U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Context-Aware Personalization for Mobile Multimedia

Abayomi Moradeyo Otebolaku

A Thesis submitted to the Faculty of Engineering University of Porto in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in

Telecommunications

Supervisor: Prof. Maria Teresa Andrade, PhD

May 2015

Context-Aware Personalization for Mobile Multimedia

Abayomi Moradeyo Otebolaku

A Thesis submitted to the Faculty of Engineering, University of Porto in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Telecommunications

Supervisor: Prof. Maria Teresa Andrade, PhD

May 2015

Context-Aware Personalization for Mobile Multimedia

By

Abayomi Moradeyo Otebolaku

A Thesis submitted to the Faculty of Engineering, University of Porto in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Telecommunications

Jury

Professor Dr. Pedro Henrique Henriques Guedes de Oliveira (President)

Professor Dr. Luís Alberto da Silva Cruz

Professor Dr. Paulo Jorge Lourenço Nunes

Professor Dr. Adriano Jorge Cardoso Moreira

Professor Dr. José Nuno Teixeira de Almeida

Professor Dr. Maria Teresa Andrade (Advisor)

May 21, 2015

Prof. Dr. Maria Teresa Andrade (Advisor)

To my late paternal grandfather

Declaration

I certify that this thesis, and the research it refers, are the product of my work, and that any ideas or quotations from the work of other people, published or otherwise, have been fully acknowledged in accordance with the standard referencing practices of the discipline, with a list of references provided in the bibliography. Similarly, the work reported in this thesis has not been submitted for any other academic award.

Abstract

The last decade has witnessed the proliferation of multimedia-enabled mobile devices and an unprecedented escalation of the amount of online multimedia content. In fact, the consumption of audiovisual content in diverse mobile platforms has risen exponentially in the last years and this trend is expected to continue in the next few years. According to a study recently published by Cisco[1], in 2013 mobile users consumed on average 2 hours of video and 2 hours of audio per month, and those numbers are expected to rise respectively to 20 and 10 hours by 2018. Users are thus increasingly accessing content anytime, anywhere and anyhow.

However, alongside this freedom to access content comes also additional entanglement of finding those that best serve the user's interests, within the maze of available content and sources. All users, irrespective of their devices, share this *herculean* task of finding the right content, but mobile users experience more frustrations. This can be explained by the fact that usually the time they have available to search, select and consume content is limited, often intermittent and shared among other tasks. In addition, because mobile users can move and perform different activities thus, the content of their interest, in format, either modality or semantics, may change accordingly. In other words, the situations of the mobile user, including the activity or task he/she performs, which can be referred to as *usage contexts*, will have an impact on the type of resources he/she will be interested in.

In this scenario of diverse users' demands and large content offers, solutions have been developed and are still being investigated to help users to select content meeting their needs. In practice, these solutions rely on personalized recommendation and content adaptation technologies.

In this thesis, we argue that more advanced solutions are still necessary, notably to satisfy distinctive requirements and characteristics of mobile users whilst being more pervasive and less intrusive. Accordingly, we believe that an approach specifically designed for mobile environments that takes into account the usage contexts and conditions to provide personalized access to content will significantly improve the quality of experience of mobile users, thus increasing their levels of satisfaction and expectation.

We have therefore proposed to investigate advanced forms of realizing such context-aware personalization by taking advantage of mobile devices' in-built sensors to acquire information concerning usage contexts, alongside monitoring user's consumptions to build innovative and implicit user profiles. Being in possession of such gathered data and employing data mining and semantic-based techniques to derive additional knowledge, the proposed system provides an innovative solution that can seamlessly or on-demand take decisions to deliver a set of recommendations to mobile users, adapted to their current *usage context* (e.g. activity the user performs; characteristics of the terminal; time of the day; location; network connection; environmental conditions, etc.). Additionally, it proposes the ability to decide whether or not the resource selected by the user from the recommended set of items, needs to be adapted to satisfy network or terminal constraints.

Essentially, these two-decision abilities of the system can be seen as its capability to address two different types of adaptation: a semantic adaptation, whereby the universe of candidate content is adapted into a limited sub-set, tailored according to the user's preferences under the specific inferred usage context; and, a syntactic adaptation, whereby the parameters of the content and/or of its encoding scheme are manipulated to comply with existing technical constraints.

The result of this work is the development of a conceptual framework for mobile context-aware multimedia personalization, designated as "*C*ontext-*A*ware Personalized *M*ultimedia *R*ecommendations for Mobile Users" (CAMR) and its prototype instantiation serving as a proof-of-concept in real world operating conditions. Experimental

[1] Cisco VNI 2014, "Cisco Visual Networking Index: Forecast and Methodology, 2013–2018". Accessible at http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html (last accessed in: Nov 17, 2014).

evaluation of CAMR shows that the proposed solution is feasible and that it can effectively provide context-aware personalized recommendations.

Main innovative features of such framework are: a) acquisition of low-level contextual data using mobile device in-built sensors (notably, accelerometer, orientation, GPS, light, sound, etc.) in a dynamic and transparent way without the need for the user to intervene; b) derivation of higher-level information concerning the context of usage (namely, the activity the user performs, or his/her location) by using classification and reasoning mechanisms; c) monitoring the user's consumptions, establishing relationships in a systematic way with the inferred usage contexts, to build implicit contextualized user profiles; d) use of the contextualized user profiles to implement a non-intrusive context-aware recommendation adopting both content-based, collaborative-based and hybrid approaches.

Consequently, the thesis realized the following contributions to advance the state of the art in mobile multimedia consumption:

1) evaluation of suitable smartphone's in-built sensors for collecting context data and for the identification of relevant features to extract and further process, aiming at simple and efficient user's activity context classifications;

2) implementation of a solution based on the previous findings, enabling accurate recognition of common types of mobile users' activity contexts;

3) development of a semantic data model to capture contextual concepts and apply reasoning procedures through a set of defined rules; such semantic mechanisms enable to derive additional contextual knowledge, notably a rich, high-level description of the situation the mobile user is in when consuming multimedia content;

4) definition of contextualized user profiles using a flexible and extensible data model and collecting data in an implicit way, without the need for the user's intervention;

5) design of a content recommendation approach using the above described contextualized user profiles, thus adapting the type of recommendations provided according to the contextual situations of the user;

6) definition of the CAMR conceptual framework for context-aware media recommendations, incorporating the concepts developed and findings achieved above;

7) systematic design and development of a system applying the principles of CAMR's conceptual framework, supporting the delivery of context-aware personalized multimedia recommendations in real-world mobile environments.

Resumo

Na última década assistimos à proliferação de dispositivos móveis com capacidades de reprodução multimédia e a um aumento sem precedentes da quantidade de conteúdo disponível on-line.

De facto, o consumo de conteúdo audiovisual numa variedade de plataformas móveis cresceu exponencialmente nos últimos anos, sendo de esperar que esta tendência se mantenha nos próximos anos. De acordo com um estudo publicado recentemente pela Cisco[2], em 2013 os utilizadores móveis consumiram em média 2 horas de vídeo e 2 horas de áudio, estimando-se que esses números subam para respectivamente 20 e 10 horas até 2018. O utilizador está assim cada vez mais a consumir conteúdos em qualquer instante, em qualquer lugar e de qualquer forma.

No entanto, toda esta facilidade em aceder a qualquer tipo de conteúdo, traz consigo a dificuldade acrescida em encontrar e seleccionar o conteúdo que efectivamente vai ao encontro dos interesses e necessidades do utilizador. Embora esta dificuldade seja partilhada por qualquer consumidor, a experiência é sem dúvida mais complicada e frustrante para os utilizadores móveis. Tal pode ser explicado pelo facto de terem disponibilidade de tempo mais limitado, intermitente, o qual é muitas vezes partilhado por outras tarefas. Mas também devido ao facto dos utilizadores móveis terem a possibilidade de se moverem e de exercerem diferentes actividades, de forma que os conteúdos de interesse, quer em formato, modalidade ou em semântica, podem ir variando consoante essa actividade. Por outros palavras, a situação em que o utilizador móvel se encontra quando está a consumir conteúdos multimédia, a qual pode ser referida como contexto de utilização e que inclui a actividade que está a exercer, irá ter um impacto significativo no tipo de recursos em que o utilizador irá estar interessado.

Neste cenário de grande diversidade de procura e quantidade de oferta, começaram a ser desenvolvidas soluções para auxiliar o consumidor a procurar e seleccionar o conteúdo adequado às suas necessidades. Na prática, tais soluções são baseadas na utilização de motores de recomendação e de adaptação de conteúdos.

Nesta tese, nós afirmamos que são necessárias soluções mais avançadas, em particular que sejam capazes de satisfazer as necessidades específicas dos consumidores móveis, sendo o menos invasivas possível. Dessa forma, acreditamos que uma solução especificamente desenvolvida para ambientes móveis, a qual entre em linha de conta com as condições do contexto de utilização para oferecer um acesso e consumo personalizado a conteúdos multimédia, será capaz de melhorar significativamente a qualidade de experiência dos utilizadores, indo ao encontro das suas expectativas e aumentando o seu grau de satisfação.

Assim, propusemo-nos investigar formas inovadoras para implementar esse tipo de personalização sensível ao contexto de utilização. A nossa abordagem passa por tirar partido de sensores existentes nos dispositivos móveis para obter informação sobre o contexto de utilização e de capacidade de processamento para monitorizar os consumos do utilizador e assim construir de modo implícito perfis de utilizador contextualizados. Essa informação recolhida é então processada através de técnicas de aprendizagem máquina, de raciocínio e semânticas, para extrair conhecimento adicional e assim permitir obter um sistema inovador que de uma forma transparente ou a pedido, toma decisões sobre o conteúdo a recomendar ao utilizador, adaptadas ao contexto de utilização (actividade exercida pelo o utilizador; características do seu terminal; a altura do dia, o dia da semana, a altura do ano, etc.; a sua localização; a ligação de rede; as condições ambientais; etc.). Para alem disso, a solução proposta tem a capacidade de decidir se o recurso seleccionado pelo utilizador da lista de recomendações que lhe foi proposta, necessita de ser adaptado para satisfazer limitações de rede e/ou terminal.

[2] Cisco VNI 2014, "Cisco Visual Networking Index: Forecast and Methodology, 2013–2018". Acessível em http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html (data do último acesso: Nov 17, 2014).

Na prática, estas duas capacidades de decisão do sistema podem ser vistas como visando dois tipos de adaptação: semântica, pela qual o universo do conteúdo candidato é adaptado num sub-conjunto formatado de acordo com as preferências do utilizador quando se encontra num determinado contexto; sintática, pela qual os parâmetros do conteúdo e/ou do esquema de codificação são manipulados para satisfazerem restrições técnicas existentes.

O resultado desta investigação consistiu na especificação de uma plataforma conceptual para recomendações sensíveis ao contexto de utilização em ambientes móveis, designada de "Context-Aware Personalized Multimedia Recommendations for Mobile Users" (CAMR) e na sua instanciação sob a forma de protótipo servindo como prova do conceito em condições reais de utilização.

As principais características inovadoras desta plataforma são: a) aquisição de dados de contexto de baixo nível utilizando os sensores do próprio dispositivo móvel (acelerómetro, orientação, GPS, luz, som) de uma forma dinâmica e transparente para o utilizador; b) obtenção de conhecimento adicional de alto nível sobre o contexto, nomeadamente a actividade exercida pelo utilizador, utilizando técnicas de classificação e raciocínio; c) monitorização dos consumos do utilizador, estabelecendo de uma forma sistemática relações com o contexto de utilização de alto nível e criando perfis de utilizador contextualizados; d) utilização dos perfis contextualizados para implementar um motor de recomendação não invasivo, sensível ao contexto de utilização seguindo quer uma abordagem baseada no conteúdo quer híbrida.

Em consequência, as contribuições mais relevantes que esta tese trouxe para o avanço do estado da arte no domínio das aplicações multimédia em ambientes móveis podem ser identificadas como sendo as seguintes:

1) avaliação dos sensores mais adequados para recolher informação de contexto e identificação das características mais relevantes a extrair e processar para identificar de forma eficiente e simples actividades físicas do consumidor móvel;
2) implementação de uma solução baseada nos resultados anteriores, capaz de reconhecer com precisão actividades físicas comuns;
3) desenvolvimento de um modelo de dados semântico para capturar conceitos de contexto e aplicar regras de raciocínio, através das quais é possível obter conhecimento adicional de alto nível, sobre o contexto de utilização.
4) Definição de um modelo flexível e extensível de perfis contextualizados, capturando dados de um modo implícito, sem a necessidade de intervenção do utilizador;
5) Projecto de um motor de recomendação de conteúdos usando os perfis contextualizados desenvolvidos e fornecendo uma lista de recomendações adaptada ao contexto de utilização;
6) Definição de uma plataforma conceptual para recomendação de recursos multimédia sensível ao contexto de utilização em ambientes móveis (CAMR) baseada nos conceitos e resultados descritos acima;
7) Projecto sistemático e desenvolvimento de uma infra-estrutura de software baseada nos princípios da plataforma conceptual CAMR, implementando aplicações personalizadas e sensíveis ao contexto de utilização em ambientes móveis.

Acknowledgments

I had mixed feelings about leaving my loved ones as I set out to start my doctoral study in faraway and unfamiliar country. I was excited about getting into a PhD but at the same time, I had no idea of what lay ahead. However, looking back, I would say that pursuing a PhD in this wonderful country and in particular in this lovely city, was a worthwhile decision!

In this regard, I would like to express my sincere gratitude to those who have made this journey a possibility, from the beginning to the end.

Foremost, my inestimable and deepest appreciation goes to my thesis advisor, Prof. Dr. Maria Teresa Andrade. She provided me with a rare opportunity to realize my dream by choosing to be my thesis advisor. Her invaluable and unique guidance, encouragement, advice and inspiration motivated me to believe that this work is possible. She ensured not only the quality technical guidance of the thesis, but also adequate funding during the years of this project. I feel extremely honored to have been her PhD student, with her immense contributions, which made this thesis a reality.

I would also like to express my gratefulness to the MAP-TELE scientific committee for giving me the privilege to enroll in the MAP-TELE PhD program and for providing a wonderful international platform of high standard to acquire advanced knowledge in telecommunications.

Special thanks to my host institution, INESC TEC, which has been my second home from the first day I arrived in Portugal, and for providing me with a unique and excellent working environment and equipment to execute this project. In addition, I express my invaluable appreciation for the financial support provided to pursue my PhD. I also appreciate all the wonderful staff and professors at INESC TEC, especially those at the center for telecommunications and multimedia (CTM), who have made my stay a huge success.

I would like to express my profound gratitude to the Portuguese Government's Foundation for Science and Technology (FCT) for providing me with a four-year PhD grant (Ref: SFRH / BD / 69517 / 2010), which was a tremendous financial assistance without which it would have been difficult to complete this doctoral study.

Additionally, my deep appreciation goes to the faculty of Engineering, University of Porto for the financial support they offered to attend international conferences.

My sincere gratitude also goes to anonymous reviewers of papers published from this thesis whose valuable input and feedback including suggestions and constructive critiques improved the technical content of the thesis. Similarly, I would like to express my deep gratitude to distinguished professors who found the time out of their extremely busy schedules to participate as members of my defense jury, and for their insightful comments to improve the quality of the thesis.

I am happy to express my heartfelt thanks to all my MAP-TELE colleagues and friends, as well as those at the Faculty of Engineering, University of Porto, and at INESC TEC for their companionship and supports during my doctoral work. In addition, sincere thanks to all my friends around the globe with whom I have shared many life experiences… your thoughts will always stay with me!

Huge gratitude to my father, Joseph Otebolaku and to my mother, Victoria Otebolaku for their patience, prayers and moral support throughout my study period. This section would not end without expressing deep gratitude to my late grandfather, Gabriel Otebolaku who was my mentor and friend, and my grandmother Dorcas Otebolaku who brought me up. Together they taught me that without education life has no meaning! For this reason, I dedicate this thesis to you both for being there when it mattered most.

Above all, deepest appreciation goes to Comfort, my companion, friend and wife. Words cannot express my gratitude for everything you have done especially for taking care of our daughters almost single-handedly while I was deeply engaged in this PhD, having limited time to spare. My deepest appreciation for your prayers and counsel as well as for standing by me in these tough years of doctoral work: When it seemed I had no friends, you stayed!

Last but certainly not the least, deepest love to my beautiful and much cherished daughters, Ife and Carisa, who were born during my doctoral work… you are both my bundles of joy.

Muito obrigado!

Abayomi Otebolaku

May 2015, Porto, Portugal.

Acronyms

ADTE	Adaptation Decision Taking Engine
ANN/NN	Artificial Neural Network/Neural Network
ARFF	Attribute Relation File Format
CACBR	Context-Aware Content Filtering based Recommendations
CACR	Context-Aware Collaborative Recommendations
CAHR	Context-Aware Hybrid Recommendations
CAMR	Context-Aware Media Recommendations
CAMRS	Context-Aware Media Recommendation Service
CARS	Context-Aware Recommendation Systems
CBF	Content Filtering Based Recommendation
CBR	Case Based Reasoning
CC/PP	Composite Capability/Preference Profiles
CDAA	Context Data Acquisition Application
CF	Collaborative Filtering based Recommendation
CIM	Context Inference Management
CM	Context Management
CUPM	Contextual User Profile Management
CUPS	Contextual User Profile Service
DAB	Digital Audio Broadcasting
DIA	Digital Item Adaptation
DT	Decision Tree
DVB	Digital Video Broadcasting
EJB	Enterprise JavaBeans
GUI	Graphical User Interface
IMEI	International Mobile Station Equipment Identity
iOS	iPhone OS
JPA	Java Persistence API
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbor

LibSVM	Library Support for Vector Machine
LogReg	Logistic Regression
MAE	Mean Absolute Error
MBCF	Memory Based Collaborative Filtering
MF	Matrix Factorization
MLP	Multilayer Perceptron
MPM	Media Profile Management
NB	Naïve Bayes
OS	Operating System
OWL	Web Ontology Language
PART	Projective Adaptive Resonance Theory
PCA	Principal Component Analysis
POI	Point of Interest
RF	Random Forest
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RM	Recommendation Management
RMSE	Root Mean Squared Error
RS	Recommendation Systems
SMO	Sequential Minimal Optimization
SOS	Some of Squares
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SWRL	Semantic Web Rule Language
UAProf	User Agent Profile
UED	Usage Environment Description
UFM	User Feedback Management
UMA	Universal Multimedia Access
UMTS	Universal Mobile Telecommunications Systems
URI	Universal Resource Identifier
URL	Universal Resource Locator
VSM	Vector Space Model
WiMAX	Worldwide Interoperability for Microwave Access
ZC	Zero Crossing

"*It always seems impossible until it is done*."

Nelson Rolihlahla Mandela

Table of Contents

Declaration vi
Abstract vii
Resumo ix
Acknowledgments xi
Acronyms xii
Table of Contents xv
List of Figures xxii
List of Tables xxv
Chapter 1: Introduction 1
1.1 Motivation 1
1.2 Thesis hypothesis formulation, objectives and contributions 5
1.2.1 Thesis objectives 5
1.2.2 Thesis contributions 6
1.3 Thesis scope and assumptions 8
1.4 Thesis structure 8
Chapter 2: Background and State of the Art 9
2.1. Overview of context-aware mobile recommendations 9
2.2 Multimedia recommendations: traditional approaches 10
2.2.1 Content-based recommendation processes 11
2.2.1.1 Content analysis 12
2.2.1.2 User profiling 12
2.2.1.3 Filtering process 13
2.2.1.4 User feedback 13
2.2.1.5 CBF's limitations 13
2.2.2 Collaborative-based recommendation processes 13
2.2.2.1 Memory based collaborative processes 14
2.2.2.2 Model based collaborative process 15
2.2.2.3 General limitations of CF systems 15
2.2.3 Hybrid recommendation processes 15

2.2.4 Context-aware recommendations 18
2.2.5 Recommendation system algorithms 19
2.2.5.1 k-Nearest Neighbor (kNN) recommendations 19
2.2.5.2 Rule based recommendations 20
2.2.5.3 SVM-based recommendations 20
2.2.5.4 Bayesian based recommendations 21
2.2.5.5 Neural network based recommendations 21
2.2.5.6 Case-based reasoning (CBR) recommendations 22
2.2.5.7 Matrix factorization based recommendations 23
2.2.6 Techniques for evaluating traditional recommendation systems 23
2.2.6.1 Experimental settings 24
2.2.6.2 Evaluation metrics 25
2.2.7 Summary of traditional recommendation algorithms 26
2.3 Modeling context information for personalization 27
2.3.1 Context definitions and concepts 27
2.3.1.1 Low-level contexts 27
2.3.1.2 High-level context information 28
2.3.1.3 Contextual situation 28
2.3.2 Context recognition 28
2.3.2.1 Context recognition overview 28
2.3.2.2 Recent context/activity recognition efforts 31
2.3.3 Context modelling and inference 33
2.3.3.1 Context representation requirements 33
2.3.3.2 Context representation models 34
2.4 User profiling 36
2.4.1 Traditional user profiling processes 36
2.4.1.1 Profile data collection 37
2.4.1.2 Traditional profile representation 37
2.4.2 Context-based user profiling 38
2.4.2.1 Contextual pre-filtering profiling 39
2.4.2.2 Contextual post-filtering profiling 39
2.4.2.3 Multidimensional contextual profiling 40
2.5 Content adaptation in mobile environments 41
2.6 Context-aware mobile media recommendation systems 43

2.7 Summary 46
Chapter 3: Context-Awareness Framework 48
3.1 Introduction 48
3.2 Motivation 48
3.3 Contextual framework development 50
3.3.1 Context sensing and recognition 51
3.3.2 Context representation 52
3.3.3 Ontological reasoning and inference 52
3.3.4 Contextual knowledge base 52
3.4 Mobile device context monitoring and recognition layer 53
3.4.1 Context recognition methodology 53
3.4.1.1 Mobile phone built-in sensors for data collection 54
3.4.1.2 Context awareness 58
3.4.2 Context model building, training, and classification 61
3.5 Context representation and ontological model 64
3.5.1 MPEG-7 and MPEG-21 metadata representations 65
3.5.2 Ontology model: the knowledge-based approach 69
3.5.2.1 Ontology preliminaries 69
3.5.2.2 The context ontology model 70
3.6 Summary 77
Chapter 4: Conceptual Framework for Context-Aware Multimedia Personalization 79
4.1 Introduction 79
4.2 CAMR requirements 80
4.2.1 Context recognition and representation 80
4.2.2 User profiling 80
4.2.3 Media classifications and recommendations 80
4.2.4 Content adaptation 81
4.3 The goal and objectives of the conceptual framework 81
4.3.1 Objectives 81
4.3.2 Basic framework assumptions 81
4.3.3 Framework development principles 82
4.3.4 CAMR high-level architecture 82
4.4 Contextual user profile 83
4.4.1 User profile definition 84
4.4.1.1 Contextual user profile representation 86

4.4.1.2 Contextual user profile formal definition 87
4.4.2 Obtaining contextual user preferences 88
4.5 Context-aware personalized media recommendations 93
4.5.1 Context-aware content-based recommendations 93
4.5.1.1 Context-aware vector space model (VSM) for content-based recommendations 94
4.5.1.2 Contextual user profile vector 95
4.5.1.3 Multimedia content profiles 96
4.5.1.4 Obtaining the similarity between a target user and candidate media item 96
4.5.2 Context-aware collaborative recommendations 98
4.5.3 Context-aware hybrid recommendations 101
4.6 Recommendation presentation adaptation service 104
4.7 Summary 105
Chapter 5:Design and Implementation 107
5.1 Introduction 107
5.2 CAMR design and development processes 107
5.3 Usage scenario 108
5.4 CAMR high-level conceptual design 110
5.4.1 CAMR user and CAMR client service 111
5.4.2 The content provider 111
5.4.3 Service Provider 111
5.4.4 CAMR Administrator 111
5.5 CAMR Functional specification 111
5.5.1 System use case model 112
5.5.2 Use case model specifications and event flow 113
5.6 Design and implementation of context-aware media recommendation framework 120
5.6.1 Requirements analysis and design decisions 120
5.6.1.1 Functional requirements 121
5.6.1.2 Non-functional requirements 123
5.6.2 CAMR basic services 124
5.6.2.1 Context management service 124
5.6.2.2 Contextual user profile management service 125
5.6.2.3 Media profile and recommendation management 126
5.6.2.4. Recommendation presentation service 126
5.6.2.5. Recommendation adaptation service 127
5.7 CAMR detailed functional architecture 127

5.7.1 CAMR layered architecture 127
5.7.2 Detailed descriptions of CAMR services 127
5.7.2.1 Recommendation management 129
5.7.2.2 Media profile management 130
5.7.2.3 Adaptation management 130
5.7.2.4 Contextual user profile management 130
5.7.2.5 User feedback management 131
5.7.2.6 Presentation management 131
5.7.2.7 Context management 131
5.7.2.8 Communication management 132
5.7.2.9 External service management 132
5.7.2.10 Client service 132
5.7.3 CAMR sequence model 132
5.7.3.1. Context recognition and classification 133
5.7.3.3. Recommendation sequence model 137
5.7.3.4. User feedback and update sequences. 139
5.7.3.5. Interactions for optional recommendation adaptation 140
5.7.4 Client subsystem implementation 141
5.7.5 Server subsystem design 147
5.7.5.1. RecommendationManager package 149
5.7.5.2. UserProfileManager package 150
5.7.5.3 The MediaProfileManager package 151
5.7.5.4. The ContextManagement Package 152
5.7.5.5. The Context Inference Management 153
5.7.5.6. The UserFeedbackManagement 154
5.7.5.7. The MediaPresentationManager 154
5.7.5.8. AdaptationManager 155
5.7.5.9 The CommunicationManager 156
5.8 CAMR framework integration, packages and possible implementation 157
5.9 Summary 159
Chapter 6: Experimental Evaluation 160
6.1 Introduction 160
6.2 Challenges of developing and evaluating context-aware media recommendations 160
6.2.1 Aspects to consider for evaluating context-aware media recommendation systems 161

6.2.2 Evaluation scope 162
6.3 Context recognition evaluation 163
6.3.1 Context recognition evaluation objectives 163
6.3.1.1 Suitability of handheld device's built-in sensors 163
6.3.1.2 Evaluation of various classification models 163
6.3.1.3 Evaluation of a combination of time series features 164
6.3.2 Context recognition experimental setup 164
6.3.2.1 Choice of mobile platform and sensors 164
6.3.2.2 The taxonomy of contexts to be recognized 165
6.3.2.3 The context data acquisition process 165
6.3.2.4 Context recognition evaluation metrics 167
6.4 Context recognition experiments 169
6.4.1 Suitability of mobile device sensor for context recognition 169
6.4.2 Feature selection evaluation 170
6.4.3 Recognizing basic activity contexts 171
6.4.4 Recognizing activity and corresponding location contexts 172
6.4.5 Context recognition using rotation vector, orientation and accelerometer data 174
6.4.6 Performance using independent test data 176
6.4.7 Real time context recognition evaluation 176
6.4.8 Context Recognition evaluation summary 179
6.5 Context-aware media recommendation evaluations 179
6.5.1 Context-aware recommendation application 179
6.5.2 Evaluation metrics 182
6.5.2.1 Classification accuracy 182
6.5.2.2 Predictive accuracy 184
6.5.3 Experimental design and evaluation 184
6.5.4 Contextual user data acquisition 185
6.5.5 Context-aware multimedia content classification evaluation 186
6.5.5.1 Data driven context-aware recommendation evaluations 186
6.5.5.2 General performance using data driven evaluation 189
6.5.5.3 Performance of context based group recommendations 190
6.5.5.4 Evaluation of new user recommendations 191
6.5.5.5 User centered evaluation 192
6.5.5.6 Context-aware recommendation application power consumption evaluation 194
6.5.5.7 Recommendation time 195

6.6 Discussion 196
6.7 Summary 198
Chapter 7: Conclusions and Future Work 199
7.1 Conclusions and contributions 199
7.2 Areas of future work 202
Bibliography 204
Appendix A 212
CAMR Client Evaluation Screenshots 212
Appendix B 215
Context Recognition Accuracies of Simple and Complex Activities of Various Classification Model 215

List of Figures

Figure 1.1- Context-aware multimedia consumption scenario 3
Figure 1.2- High-level architecture of the proposed solution 5
Figure 2.1- A typical traditional recommendation process 10
Figure 2.2- Traditional recommendations(a) Context-aware recommendations (b) 18
Figure 2.3- A three layer information flow to obtain high-level contexts from sensor's raw data 29
Figure 2.4- A generic architecture of context recognition processes 29
Figure 2.5- User profiling process loop 36
Figure 2.6 -Contextual pre-filtered profiling 39
Figure 2.7 - Contextual post-filtering 40
Figure 2.8- Model based contextual profiling 40
Figure 2.9- Typical media adaptation architecture 41
Figure 3.1- High-level architecture of the context awareness framework 49
Figure 3.2- Detailed architectural view of the proposed context awareness framework 51
Figure 3.3- High-level view of context recognition process cycle 53
Figure 3.4- Various embedded sensors in modern phone 54
Figure 3.5- Coordinate system relative to a device used by Android Sensors 55
Figure 3.6 - Low-level signals of some user activities obtained from mobile phone embedded sensors 58
Figure 3.7 - Context data acquisition application (CDAA) 64
Figure 3.8 - MPEG-21 UEDs concepts 65
Figure 3.9- Terminal capability UED 66
Figure 3.10 - Network characteristic UED 67
Figure 3.11 - User information UED 67
Figure 3.12 -MPEG-21-user information UED schema extension for user activity 68
Figure 3.13- Natural environment characteristics UED 69
Figure 3.14- Context inference model 71
Figure 3.15 - Context ontology mode 71
Figure 3.16 -Ontology model 72
Figure 3.17 - User concept 72
Figure 3.18 - Device concept 73
Figure 3.19 - Environment concept 73
Figure 3.20 - Media concept 74
Figure 3.21 - Network concept 74
Figure 3.22- Activity concept 75
Figure 3.23- Time concept 76
Figure 3.24 - Example ontology instantiation 76
Figure 3.25 - Listings 1, 2 and 3: Example rules for inferring higher-level contextual information 77
Figure 4.1- Multimedia consumption via smart devices in mobile environments 79
Figure 4.2- CAMR conceptual frameworks showing some of its basic functional parts 83
Figure 4.3-Proposed framework process cycle 83
Figure 4.4- Contextual user profiling model 85
Figure 4.5- The generic contextual user profile representation 86
Figure 4.6- An example contextual user profile representation for movie, music and news content 87
Figure 4.7- Retrieving similar contexts from user profile 90
Figure 4.8- Example user profile representation for movies. 92
Figure 4.9- Hypothetical user profile vector. 92

Figure 4.10-A traditional recommendation process....94
Figure 4.11- A generic context-aware content-based recommendation process....94
Figure 4.12- Excerpt of hypothetical MPEG-7 metadata....96
Figure 4.13 - Hypothetical candidate media vector....97
Figure 4.14- Context-aware content-based recommendation process....97
Figure 4.15- CAMR context-aware content-based recommendation process workflow....98
Figure 4.16- A generic context-aware collaborative filtering based recommendation .process....99
Figure 4.17- CAMR context-aware collaborative filtering process workflow....99
Figure 4.18- Context-aware collaborative process....101
Figure 4.19- A generic architecture for context-aware hybrid recommendation....103
Figure 4.20- CAMR context-aware hybrid recommendation process workflow....103
Figure 4.21- Context-aware content based collaborative process....104
Figure 5.1- CAMR development process....107
Figure 5.2- Summary of the activities of the proposed system for realizing the usage scenario....109
Figure 5.3- High-level conceptual architecture of CAMR showing its primary actors and subsystems....110
Figure 5.4- CAMR use case model showing functional requirements of the proposed framework....113
Figure 5.5- CAMR layered architecture....128
Figure 5.6- CAMR functional architecture....129
Figure 5.7- Simplified sequence of interactions in context recognition....134
Figure 5.8- Simplified sequence of interactions for user preference inference....136
Figure 5.9- Simplified sequence of interactions during contextual user profiling....136
Figure 5.10- Simplified sequence of interactions for an adaptive context-aware recommendation process....138
Figure 5.11- Simplified sequences of interactions for user feedback and contextual user profile update....140
Figure 5.12- Sequence of interactions during recommended content adaptation....141
Figure 5.13- Analysis model and package references of the client subsystem of the proposed system....142
Figure 5.14- Aactivity model that captures the context processing on the user device....142
Figure 5.15- Client service interface showing resources that are exposed by the web service....145
Figure 5.16- POST request for user's contextual information....146
Figure 5.17- GET request for user's contextual information....146
Figure 5.18- High-level architecture of CAMR server subsystem....148
Figure 5.19- Detailed UML model of CAMR server subsystem of the proposed framework....148
Figure 5.20- Excerpt from the recommendation manager class model showing its most important classes....149
Figure 5.21- Simplified class model of the user profile manager 151
Figure 5.22- Media profile manager simplified class model....152
Figure 5.23- Excerpt of context manager class model....153
Figure 5.24- Context inference manager model based ontological reasoning and inference....153
Figure 5.25- *FeedbackManager's* simplified class model....154
Figure 5.26- Illustrates the interactions between client and the server mediated by the *CommnicationManager*....156
Figure 5.27- CAMR High-level software and hardware platform-independent integration....158
Figure 5.28- CAMR package intra and extra communications between various modules....158
Figure 6.1- User context data collector interfaces....165
Figure 6.2- Logged low-level smartphone sensor data....165
Figure 6.3- Logged low-level smartphone sensor data....166
Figure 6.4 - Logging low-level context....166
Figure 6.5 - Classifiers' recognition RMSE for selected window lengths....170
Figure 6.6 - Window length versus context recognition performance....170
Figure 6.7 - Classifiers' FScore for selected activities....171
Figure 6.8 - Classifiers' performance for activity with and without location recognition....173
Figure 6.9 - Recognition evaluation using accelerometer, orientation, and rotation vector data independently....175

Figure 6.10 - RMSE Comparison of Sensor Combination. 176
Figure 6.11 - User independent evaluation of classification models for recognition of activities 177
Figure 6.12 - Context browser showing some selected real-time recognized contexts and activities 177
Figure 6.13 - Real-time recognition accuracy of some basic user activity contexts 178
Figure 6.14 - CAMR running on Galaxy S I9000 device 180
Figure 6.15- User preference setting....180
Figure 6.16 - User preference setting....180
Figure 6.17 - Evaluation interface and recommendation Interface 181
Figure 6.18 - Recommendation presentation interface 181
Figure 6.19 - Recommendation interface playing recommended movie clip 181
Figure 6.20 - Definition of precision and recall in the context of recommendation systems....182
Figure 6.21– Computing precision, recall and f-score for recommendation system 183
Figure 6.22- Controlled experimental setup 186
Figure 6.23- Comparison of collaborative (a) and context-aware collaborative (b) based recommendations....187
Figure 6.24- Content-based and (a) context-aware content based (b) recommendation precisions....188
Figure 6.25- Performance combining context-aware hybrid process 188
Figure 6.26- Comparisons of context-aware processes (CACR, CACBR and CAHR)....189
Figure 6.27- Average precisions for all users 190
Figure 6.28- Using contextual information of group of users for content recommendations. 190
Figure 6.29- Accuracy of recommendations using a new user's contexts compared to non-contexts....191
Figure 6.30- Average precision from test users 'feedback 193
Figure 6.31- Average recall from test users' feedback 194
Figure 6.32- Recommendation time versus number of candidate media items 196

List of Tables

Table 2.1 - Weaknesses of CBF, CF and hybrid recommenders 18
Table 2.2- Comparison of context models 34
Table 3.1- Feature descriptions and definitions 61
Table 4.1- User profile definition 87
Table 4.2- Example context information in a typical user profile preferences 90
Table 4.3- Example similarity scores or categorical context features 91
Table 4.4- User contextual consumption log showing new user problem 101
Table 5.1- CAMR main use cases , descriptions and actors 112
Table 5.2 - Get contextual information use case 114
Table 5.3 -Infer high-level contextual information use case 115
Table 5.4 - Generate contextual user profile use case 115
Table 5.5 - Retrieve user preference use case 115
Table 5.6 - Predict user preference 116
Table 5.7 - Get contextual recommendation use case 116
Table 5.8 -Get Recommendations use case 117
Table 5.9 - Present recommendations use case 117
Table 5.10 - Get User Feedback use case 118
Table 5.11 - Update user profile use case 118
Table 5.12 - Manage multimedia metadata use case 119
Table 5.13 - Manage knowledgebase use case 119
Table 5.14 - Manage user use case 119
Table 5.15 - Manage content provider/service provider use cases 120
Table 5.16 - List of requirements and related use cases 121
Table 5.17 - List of CAMR services and related use cases 124
Table 5.18 -Sequence models and associated use cases 133
Table 5.19 - Selected functions of the context recognition and classification sequence 134
Table 5.20 - Selected functions of the user profiling sequence 137
Table 5.21 - Selected functions in of the recommendation sequence 138
Table 5.22 - Selected functions in of the user profile update sequence 139
Table 5.23 - Examples of REST implementation of client service resources 147
Table 5.24 - Usage environment context information 155
Table 5.25 – Content information 155
Table 5.26 – Example of adaptation decision parameters 155
Table 5.27 - Communication Manager main methods and their descriptions 157
Table 6.1 - Summary of popular algorithms used in context recognition 167
Table 6.2 - Window lengths and number of class instances 169
Table 6.3 - Activity recognition capability of each time domain feature and classification algorithms 171
Table 6.4 - KNN confusion matrix for simple activity contexts 172
Table 6.5 - SMO confusion matrix for simple activity contexts 172
Table 6.6 – KNN Confusion matrix for location and activity contexts 173
Table 6.7 – SMO Confusion matrix for location and activity context 174
Table 6.8- Comparison of various activity context recognition accuracy 178
Table 6.9- Sample contextual information used in the experimental evaluations 185
Table 6.10- Mean Average Precision and Confidence Level 192
Table 6.11 - Power consumption evaluations of the system 194

Chapter 1

Introduction

In the last years, handheld devices, such as smartphones, have become indispensable in our daily routines. Unlike traditional mobile phones, smartphones are gradually replacing our desktops as the primary computing platform, as they increasingly become more powerful in terms of processing capacity, network connectivity, and multimedia support. Nowadays, with the pervasive availability of multimedia-enabled handheld devices, it is quite common to see people in their daily routines navigating on the Web with their devices, seeking useful information or interesting content. However, the sheer volume of Web-based multimedia content can easily overwhelm mobile users, with only a small percentage actually meeting their needs and preferences. Often, because this content does not suit their preferences, users waste invaluable time searching for interesting but elusive media items. The information retrieval community generally refers to this kind of problem as ***information overload*** [1]. This problem is particularly more profound in mobile environments, considering the diversity of environments and multiplicity of activities or tasks that users perform. Their time to search for useful content maybe limited and their interests may vary according to their particular situations. Essentially, mobile users may have different content needs, depending on their contextual situations, especially their locations, environmental conditions and activities. It is therefore important to track mobile users' contexts, in which they carry out their important daily routines.

The hypothesis of this thesis is that by using contextual information, it is possible to personalize the delivery of multimedia content to mobile users according to their contextual situations, and thus effectively providing them with the content they need or prefer in that particular situation. This will contribute to minimizing their sense of frustrations when navigating through overwhelming stream of content, thus increasing their level of satisfaction. Therefore, based on this hypothesis, this thesis focuses on the conception and the development of a conceptual framework to support context-aware personalized multimedia recommendations in mobile environments. In that process, it investigates the dynamic recognition of mobile user's contexts and activities in a context-awareness framework as the backbone of a truly context-aware personalized recommendation. Additionally, it investigates how to use recognized contexts to support dynamic context-aware user profiling to enhance mobile content classification and recommendations. This chapter contextualizes the thesis, presents its motivation, goal, objectives, technical solutions, contributions. It also outlines its organization.

1.1 Motivation

Smart devices, equipped with sophisticated functional operating systems such as iOS and Android, with high-end sensing capabilities, are becoming indispensable companions in our daily lives. Their importance and roles can no longer be underestimated. First generation mobile phones traditionally were limited to only voice and text communications. However, in recent times, with substantially improved multimedia, memory, processing, and storage capabilities, mobile devices have transformed into smart devices, having in some cases more capability than traditional desktop computers. From the 1956 SRA/Ericsson Mobile Telephone System A (MTA), through the 1983 Motorola DynaTac 8000x, to the 2014 Samsung Galaxy S6 and Apple iPhone 6, among others, these devices have evolved into powerful computing machines. They can now process incredible amounts of data, which before could only be processed by high-end personal computers. Moreover, they offer other types of capabilities that are not present in desktop or even laptop computers. For example, they are now equipped with a number of built-in sensors, which possess the potential to revolutionize mobile information processing. In fact, it is now common to find built-in sensors such as GPS, accelerometers, gyroscopes, orientation, microphones, cameras, etc. in many smartphones. With these increasing and improving capabilities come exciting opportunities to explore, beyond mere voice and text

communications, namely the provisioning of advanced multimedia services that can be tailored to the user's contextual situations and needs, by taking profit of data obtained from the built-in sensors.

These developments, including factors such as the proliferation of broadband Internet access and mobile devices affordability, are enabling millions of people to possess smartphones not only for voice communication, but also more importantly for multimedia consumption and production [2]. Individuals with smartphones are now multimedia content producers and consumers, publishing and consuming huge amount of content daily on the Web, depending on their contextual situations (environments). From YouTube, Facebook, Twitter to Google+, multimedia content has now become seamlessly available. While on the move, users can access those through a variety of communication networks, namely mobile (UMTS), wireless (Wi-Fi/WIMAX) or even wireless broadcast (DVB-H, DAB DMB). Simply put, mobile devices are fast becoming the primary platform for multimedia entertainments. However, this significant progress has also brought with it many challenges. The overwhelming volume of contents, which users have to navigate endlessly, seeking relevant ones that suit their preferences and needs, has emerged as a new challenge. This challenge is a new form of "*information overload problem*" [3], which we refer to as *"mobile information/multimedia overload problem",* constitutes a major challenge [4]. Like the traditional information overload problem, it emerged from the deluge of information that is published continuously by millions of users on the Web. However, in mobile environments, it assumes a new dimension due to additional constraints. Mobile users can find themselves in a variety of situations and often are involved in multiple activities or tasks, having limited intermittent time at their disposal and having needs that may change according to those situations and conditions. Accordingly, solutions that are able to help users to find useful information, namely those based on recommendation systems, are becoming an essential piece of this puzzle. However, these solutions do not consider the peculiar situations of mobile users, where their contextual information plays important roles in the kind of information they produce or consume [5]. Consuming contents that meet the user's preferences and expectations must take into account, not only the preferences of the user, but also the situations surrounding the user when she expresses such preferences, including her device capabilities, and network conditions [6]-[7]. This is consequent to the mobile nature of smart device users whose preferences change as their contexts and activities change, requiring different multimedia content in different contexts. Let us take the following simple scenario, which is elaborated in chapter 5, as an example to illustrate this challenge.

Ana enjoys watching latest movies at the cinema, especially on dry weekends. She relies on personalized media recommendation service to provide her with favorite choices. She wants to select at least one interesting movie from the set to watch on Saturday with her friends at the cinema. While she is jogging on Friday at 18h30, the system provides her with a set of personalized recommended movies with only titles and synopsis of each movie, presented to her in auditory form or as SMS alerts. Later on that night, at 20h30 while sitting at home, she receives, on her smartphone, trailers of those movies, which she has previewed while jogging. She then watches those trailers to decide the movie she wants to see with her friends at the cinema on Saturday.

The system would be able to take decisions and consequently to select relevant content appropriate to Ana's contextual conditions in both situations ("outside, jogging, on a Friday before a rainy weekend" and "sitting at home on a Friday evening before a rainy weekend") after having collected contextual data and processed it, as well as consulted Ana's profile. It would thus be in possession of additional and richer knowledge concerning Ana's situation. Such richer knowledge could be hypothetically:

- Ana likes to go to the movies during dry weekends when she is in her hometown;
- Ana likes to go cycling with her friends during sunny weekends in her hometown outskirts;
- Ana is jogging on a Friday evening;
- Ana is seated at home;
- Ana is in her hometown;
- the weather forecast announces rain for the weekend in the area where Ana lives, etc.

In the scenario illustrated above, current recommendation systems can easily provide to Ana a set of movies of her interest based on her previous choices. Such systems will neither be able to decide between sending her a set of

recommended movies with full trailers and a short list with limited information in auditory form. Nor will they be able to decide between recommending a set of beautiful tracks on the countryside that are ideal for cycling and a set of movies to watch on a dry weekend at the cinema. In fact, if Ana has never made any decision before concerning consumption, such system might fail to provide her with relevant content. Such decisions, and thus more adequate recommendations, can only be made if the system is able to sense and accurately understand the usage contexts, whilst knowing the choices Ana has made in the past under similar conditions. Figure 1.1 illustrates at high-level the overview of the system providing this kind of solutions for adaptive and context-aware personalized multimedia recommendations in a typical mobile and wireless environment.

In the last decades, significant progress has been achieved to address the traditional information overload problem of such systems, especially in the field of Universal Multimedia Access (UMA) [6]-[7]. Existing UMA based solutions, concerning multimedia consumption, have considered the challenges of delivering only explicitly requested content, taking into consideration a limited set of contextual constraints [8]. They generally overlooked the aspect of anticipating user's needs, and therefore addressing mobile information overload problem has not been their focus.

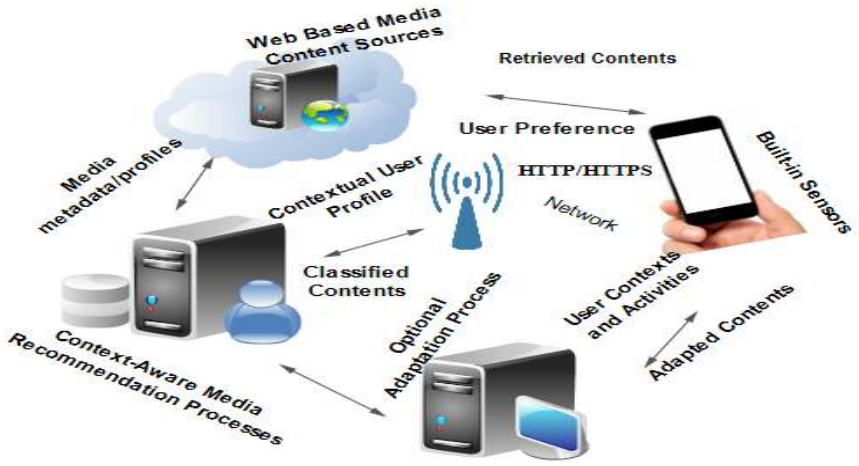


Figure 1.1 Context-aware multimedia consumption scenario

Anticipating mobile user's content needs requires an understanding of her environment situations and contexts, which generally influence the user's preferences. Additionally, a solution that integrates the user's content needs, activities, and contexts in real-time with existing traditional methods, such as content classification and recommendation to address mobile multimedia content overload problem is much more desirable.

The scientific community has been very active during the last decades investigating topics of information retrieval and recommendation techniques to address the information overload problem [9]-[23]. In this regard, personalized recommendation system techniques have been investigated and proposed as effective solutions capable of improving user multimedia consumption experience by delivering interesting and tailored multimedia content according to users' preferences [13]-[16]. These systems have the capability to select among many possible options of contents and to deliver those more relevant and interesting to users. On one hand, information retrieval systems provide the functionality that can lead a user to "*those documents that best enable him/her to satisfy his/her need for information*" [27]. They are typically useful when people already know what they want to find. On the other hand, recommendation systems assist people to make choices, either based on the opinions of other people who are close to them or based on their own previous experiences [10]-[11], [14], [23]. Personalized systems (e.g. recommendation systems) can build, manage, and represent information, customized to a particular user [27]. Personalized recommendations grew mainly out of information retrieval to minimize information overload and to retain customers, by anticipating their content needs [27]. As succinctly put by Burke in [28], the criteria of "*individualized, interesting, and useful*" distinguish recommendation systems from retrieval systems. If we remove personalization from recommendation process, we are left with only a set of popular items, which are independent of users. Personalization is the key feature of

recommendation systems, which distinguishes them from retrieval systems. Therefore, in information retrieval systems such as search engines, the results that are relevant to a query are always the same irrespective of the individual who issued such query or where and when it is issued [23], [29]. Examples of personalized systems can be found in tourism [19], [30], targeted advertisement [31], health [32], transportation [33], music [34], news [9]-[10], commerce [35]-[36], and movies [37]. However, traditional personalized recommendation systems have some limitations. One of such limitations is that they assume that user preferences are fixed, i.e. that they do not frequently change according to user's contexts [15], [22], [38]. Thus, they lack the ability to infer user's contextual preferences, especially in mobile environments, relying only on static preferences given explicitly by users to content they have previously consumed [5]. For example, in the form of ratings, or else obtained implicitly by the system by analyzing the user's consumption. In mobile environments, however, the contextual conditions that characterize the situations where consumptions of multimedia content occur assume a much more important role. In fact, mobile users usually engage in different activities compared to those of traditional desktop users and hence have different content needs, compelled by their changing activities and contextual situations.

Until recently, little attention has been paid to the importance of contextual data within that process. Partly because the objective was to solve the problem in a more traditional consumption environments such as desktop computers and TVs, the research on personalized recommendations has concentrated on developing, optimizing and evaluating traditional recommendation approaches to determine the best ones [39]. Still, more recently, researchers have effectively started to incorporate contextual information into such systems, therefore leading to the development of pioneer context-aware recommendation systems [2], [13]-[22], [26], [34], [40]-[45]. Nonetheless, those systems still present a number of other limitations, which opens up space for additional improvements, especially when considering their application in mobile environments.

The first of such main limitations comes from the fact that most of them still rely on the user to explicitly provide his/her opinion about the resource he/she has just consumed in the form of a rating [14]. Using only content ratings to predict content for mobile users can prove to be impractical given that most of the times such ratings simply do not exist. Thus, the unavailability of ratings makes such systems unreliable, leading to the cold start and sparsity problems [5]. This coldstart problem can even be aggravated when ratings are correlated to the contextual conditions under which the user has consumed the content, given the considerable variety of situations of a mobile user when consuming content.

The second limitation comes from their rather poorly efficient and considerably complex ability to recognize context[3]. Some of those systems do not even provide this kind of ability, as they simply assume contextual data is readily available. They do use contextual data but they rely on hypothetical external modules to provide them such data. Others, albeit implementing internally this feature, offer very basic functionality, limited to the use of space and time coordinates and/or characteristics of the terminal device and network connections. Finally, some of them even require the mobile user to supply the contextual data manually. Such solutions are obviously very far away from the pervasiveness and non-intrusiveness paradigm has aimed at achieving. Additionally, common to most of them is the fact that they do not have the ability to establish relations amongst the different types of gathered contextual information. This means that they are not able to derive additional, higher-level knowledge through the combination of different low-level context data, the way humans do. This may prove to be a significant limitation in what concerns their context-awareness ability, as the same low-level context may lead to different conclusions when interrelated with other information [24].

The third problem is their limited ability to address the challenge of anticipating user's needs successfully. Addressing this problem requires an efficient way to model the preferences of the user and track relative changes into a dynamic

[3] (or alternatively, "comes from the rather poor efficiency and considerable complexity of their context recognition functionality").

user profile [46]. Moreover, given that the preferences of mobile users are related to the usage context, such model must also be able to relate consistently preferences to the corresponding context. The representation of these dynamic relationships into a user profile has not yet been adequately achieved by existing solutions. Instead, they normally focus on modeling only static associations between context data and user preferences [47]. This thesis seeks to help overcome such limitations!

1.2 Thesis hypothesis formulation, objectives and contributions

In this thesis we argue that it is possible to build less intrusive and more pervasive multimedia applications specially designed for mobile users, by developing: 1) efficient context recognition engines that seamlessly use built-in sensors of mobile devices; 2) monitoring and classification capability to dynamically obtain user's preferences and systematically associate them to the usage context, into dynamic contextualized user profiles; and 3) recommendation engines that transparently or on-demand supply the user with a list of multimedia resources especially suited to his/her preferences and current situation.

1.2.1 Thesis objectives

The support for adaptive, personalized context-aware mobile multimedia content classification and recommendation requires some major processing steps:

- Acquisition of data characterizing the usage context (environmental, terminal, networks and user conditions) using physical and software of mobile device built-in sensors, such data is referred to as low-level contexts.
- Transformation of low-level context by classification and inference mechanisms into a meaningful high-level form of contextual information.
- Establishing relationships between past and current high-level context information and associated preferences to be able to anticipate the user's current preferences.
- Recommending multimedia content to mobile users, based on contextualized preferences together with the user's current contexts.
- The contextualized preferences together with the user's current context can then be used to classify and recommend multimedia content for mobile users.
- Optionally, this information can be used to adapt the presentation of the recommended multimedia content to suit mobile device and wireless network constraints

Considering the steps above, the main goal of this thesis is to develop a generic conceptual framework for context-aware personalized multimedia recommendations in mobile environments as illustrated in Figure 1.2. The specific objectives to realize the above stated goal are to:

- Develop a context awareness framework incorporating context recognition, inference, and representation, using the mobile device built-in sensors to identify user's contexts and activities.

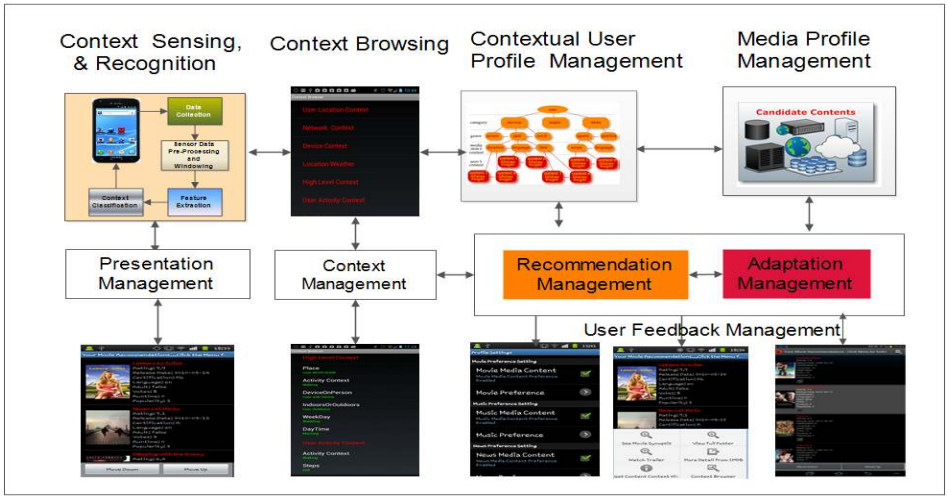


Figure 1.2- High-level architecture of the proposed solution

- Develop a contextual user profile model, which interfaces with the context awareness framework, relating the user's multimedia content preferences and their changing contexts and activities.
- Develop contextual recommendation techniques by extending the traditional approaches using the contextual user profile model and context and activity information for multimedia content classification for mobile users.
- Design and development of a system supporting context-aware personalized recommendations.
- Propose an optional adaptation decision ability of the recommended multimedia content.

1.2.2 Thesis contributions

This thesis makes the following contributions:

- A dynamic context awareness framework for mobile environment. The thesis developed a context awareness framework that supports a dynamic contextual user profile model for adaptive "*C*ontext-*A*ware Personalized *M*ultimedia *R*ecommendations for mobile users" (Context-Aware Media Recommendations, **CAMR**). CAMR supports the integration of context sensing, recognition, and inference, using classification algorithms, an ontology-based context model and user preferences to provide and personalize media items to mobile users. The developed context framework can run on smart devices, taking advantage of their built-in sensors, and using classification algorithms to identify mobile user's context information and activities. The context framework gathers low-level events from smartphone in-built sensors, such as accelerometer, orientation, location (GPS, Wi-Fi), light, orientation sensors, etc., and pre-processes them to infer high-level context information and the activity the user performs. Other low-level contexts such as characteristics of the terminal and network conditions are also monitored. Altogether, the system is able to understand the situations of the user, network and environmental conditions.
- Assessment of different types of sensors for different types of features extracted from the data acquired from those sensors and different machine-learning techniques to classify accurately the mobile user activities.
- A contextualized user profile model that unifies the context awareness framework and multimedia content classification processes. The context-sensitive user profile model supports and contextualizes recommendation techniques such as content-based filtering, collaborative filtering, and hybrid recommendations. The contextual user profile model summarizes the user's consumptions into a set of categories. Categories, genres and properties are associated to user's contextual situations. It offers a flexible structure upon which reasoning techniques can be applied to derive the preferences of the user in his/her current contextual situation.
- Context-aware classification and recommendation techniques. The traditional recommendation algorithms (content based filtering and collaborative recommendation) suggest content to a user based on opinions and preferences of other users or based on the user's own consumption history [12]. The thesis builds multimedia content recommendation and classification on top of the contextualized user profile model and context recognition framework in a flexible way to allow the contextual filtering of user multimedia content consumption preferences. This enables to obtain a context-aware content based, context-aware collaborative and context-aware content-based collaborative processes for mobile multimedia content classification and recommendation.
- Design and implementation of a mobile application adopting the CAMR principles and techniques, offering context-aware personalized recommendations to mobile users thus serving as a proof-of-concept. Being a concrete implementation of the conceptual framework applied to a specific case study (movie entertainment on mobile devices), it also served to demonstrate the feasibility of the proposed conceptual solution and to conduct experimental tests to evaluate its efficiency. First, it was used to simulate the case study and second, it was used in a user study to evaluate user satisfaction of the recommendations. The experimental evaluation results show that the approach is feasible and reliable, and that it can satisfactorily recommend contextually relevant contents to users.

In parallel, a number of peer-reviewed publications describing the contributions have been produced. Those under review are indicated.

- Journals
  - A.M. Otebolaku and M.T. Andrade “Context-Aware Media Recommendations for Smart Devices" published in Springer Journal of Ambient Intelligence and Humanized Computing, pp 13-36, Vol. 6 Issue 1, February 2015..
  - A.M. Otebolaku and M.T. Andrade "User Context Recognition Using Smartphone Sensors and Classification Models" (under review).
  - A.M. Otebolaku and M.T. Andrade “ A Context-based System for Mobile Multimedia Personalization” (under review)
- Book Chapter
  - A.M. Otebolaku and M.T. Andrade "Context-Aware Multimedia Content Recommendations for Smartphone Users” Book Chapter published in the IGI-Global 3rd Edition of the Encyclopedia of Information Science and Technology, July, 2014.
- International Conferences and Workshops
  - A.M. Otebolaku and M.T. Andrade, “Context Representation for Context-Aware Mobile Multimedia Recommendation,” In Proceedings of the 15th IASTED International Conference on Internet and Multimedia Systems and Applications, Washington, USA, 2011.
  - A.M. Otebolaku and M.T. Andrade, "Recognizing High-Level Contexts from Smartphone Built-in Sensor for Mobile Media Content Recommendation" In HumoComp 2013 workshop of the 14th IEEE International Conference on Mobile Data Management, pp. 142-147, Milan, Italy (2013).
  - A.M. Otebolaku, M.T. Andrade, "Context-Aware User Profiling and Multimedia Content Classification Smart Devices" In the proceedings of The 28th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA-2014) Victoria, Canada, pp. 560-565, May 13-16, 2014.
  - A.M. Otebolaku, M.T. Andrade " Context-Aware Media Recommendations" In the proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA-2014), Victoria, Canada, pp. 191-196, May 13-16, 2014.
  - A.M. Otebolaku and M.T. Andrade "Supporting Context-Aware Cloud-Based Media Recommendations for Smartphones" In the proceeding of the 2nd IEEE International Conference on Mobile Cloud Computing, Services and Engineering, Oxford, UK, pp.109-116, April 7-10, 2014.
  - A.M. Otebolaku and M.T. Andrade "A Context-Aware Media Recommendation Framework on Smartphones" In L.De Stricker (Ed.), In ECUMICT 2014: Proceedings of the European Conference on the Use of Modern Information and Communication Technologies, Gent, March 2014, Springer Lecture Notes in Electrical Engineering, Vol. 302, pp. 87-108, Springer International Publishing Switzerland, March 27-28, 2014.

1.3 Thesis scope and assumptions

This thesis focuses on mobile user's context awareness and adaptive context-aware personalized mobile multimedia classification and recommendations, applied in the broader field of Universal Multimedia Access (UMA) [6]-[7]. The multimedia content classifications and recommendations rely on contextual user preferences managed by a new contextual user profile model, developed on top of a context awareness framework. The initial objectives formulated for this thesis emphasized the aspects of using contexts to adapt contents rather than using contexts to recommend contents. But, events and developments in the field have overtaken this goal in the last years, and therefore has been modified and realigned with a broader scope, focusing on generic context-aware mobile multimedia content personalization framework, with optional mechanisms to adapt the presentation of personalized content to the user's smart device and network constraints. A context-aware mobile personalized recommendation application was developed on top of the framework as a proof-of-concept. The application is a concrete case study of the framework where its components have been implemented and coordinated to provide contextual movie recommendations to mobile users. The framework operates on the assumption that MPEG-21 based adaptation decision taking engine (ADTE) [6]-[7] and adaptation engine, which are responsible for tailoring the personalized content to suit the user's smartphone device constraints and network conditions, are readily available. Additionally, the thesis assumes that all Web-based multimedia content are represented based on MPEG-7 description tools [48]. Nevertheless, recommending content can also be seen as a form of adaptation, whereby the universe of candidate content is transformed into a limited set of items constrained by the user's preferences and needs. Additionally, the developed techniques are able to decide the format with which the set of recommended content should be presented to the user depending on constraints imposed by the usage context (as described in the hypothetical example of section 1.1).

1.4 Thesis structure

The thesis is structured as follows. Chapter 2 discusses the thesis background, technologies and surveys related work. Chapter 3 presents a mobile phone based context awareness framework, detailing how to obtain high-level context information from low-level smartphone built-in sensor events, using data mining algorithms. It also presents the proposal for a representation of context information using MPEG-21 and Ontology based model for semantic inference of context information, allowing reasoning based on relationships between the context information to derive better understanding of the contextual situation in real life situations. Chapter 4 presents the conceptual context-aware personalized content classification and recommendation framework, detailing the extended recommendation techniques with contextual information. The chapter also presents the contextual user profile model, which unifies the recommendation processes and the context recognition model. Chapter 5 presents systematic design of a software support for context-aware personalized media recommendations, detailing its architectural design and discusses each of its components and implementation. Chapter 6 presents experimental validation of the proposed framework. It also discusses the results of series of experiments conducted to validate the proposed solution. Additionally, it discusses conclusions drawn from those results. Finally, chapter 7 presents conclusions and future direction of the thesis.

Chapter 2

Background and State of the Art

This chapter introduces the background and related work on context-aware personalized media recommendations. It also discusses how this thesis relates to and differs from the state of the art. Section 2.1 provides an overview of context-aware mobile recommendations. In section 2.2, the chapter examines traditional recommendations and their techniques. In 2.3, it examines context-modeling approaches and discusses context, types of context, and techniques for deriving high-level contexts from sensor's low-level events. Section 2.4 presents an overview of traditional user profiling techniques, while section 2.5 presents a review of techniques for adaptive presentation of mobile multimedia. Section 2.6 presents some existing context-aware mobile multimedia recommendation systems. It reviews, summarizes, and discusses their strengths and weaknesses in comparison with the solution presented in this thesis. Section 2.7 summarizes the chapter.

2.1. Overview of context-aware mobile recommendations

Traditional recommendation approaches were designed to provide efficient and effective item recommendations from massive collections of information items. In the traditional computing environment, users often go through much difficulty to find suitable online media items due to the overwhelming number of available options, and the lack of support for making quick and effective decisions on relevant choices. This situation is otherwise known as information overload [5]. To address the *information overload* problem, recommendation systems (RSs) have been proposed, investigated, and developed as effective decision support tools, providing item recommendations and personalized services to meet user's needs and preferences at each request [15],[18],[30]. The information overload problem becomes more challenging in mobile environments due to the inherent problems of mobile computing environment, such as the user mobility related issues in tersm of space and time. Mobile users have limited time to browse through large quantity of content as they move around performing different activities. In addition, the space limitation in terms of screen size compared to desktops makes it more difficut to visualize and preview content thereby making comaprison more difficult [4], [30], [49]-[50]. Consequently, recommendation techniques, which have been highly successful for PC users cannot be straightforwardly applied in mobile environments [30]. Therefore, address the time and space limitations of personalization systems in mobile environments, it is necessary to develop recommendation techniques that can provide relevant items [4], [50].

Mobile recommendation systems are a subset of recommendation systems that aim to find relevant information items in mobile environments, satisfying user's preferences [30]. Nevertheless, the global proliferation of handheld devices such as tablets, mobile phones, and smartphones, which are continuosly becoming smarter due to the integration of sensing capabilities, e.g. GPS, accelerometer etc., sophisticated operating platforms, such as Android™ and iOS™, has fostered the development and delivery of smart services in the area of mobile and pervasive computing. And as handheld devices gradually become the primary platform for multimedia information access, combined with recommendation techniques, they are evolving as the key and effective tools in the hands of mobile users for ubiquitous entertainment platform, providing anywhere, anytime access to multimedia information such as music, videos, news, sport etc. [50]. Recently, because of these developments, there have been significant efforts to integrate recommendation techniques in mobile computing environments, especially for multimedia item recommendations. For example, Yu et al. [18] developed a mobile recommendation system to support the delivery of multimedia items relevant to contexts of mobile users. Another example is a system described in [51] that provides ubiquitous and mobile information needs such as on-site context-aware access to cultural heritage for tourists. There are several other important developments in the area of news, transport, advertisement, and healthcare [4], [29]. However, much of the

efforts in recommendation systems have concentrated on traditional computing platforms. These traditional recommendation techniques cannot be applied directly to address the information overload problem in mobile environments [30]. The application of RSs in mobile environments faces the inherent challenges such as the limitations of mobile devices and those of the wireless networks, the environmental impacts, and changing behaviors of mobile users [50], [52]. However, mobile devices provide two important characteristics that can be explored by recommendation systems. First, the ubiquity of Internet-ready handheld devices, which users always carry around anywhere they go. This provides unique opportunity for media items' accessibility anywhere, anytime. Second, because modern mobile devices now come with sophisticated operating systems, e.g. iOS, Android, etc., embedded sensors such as GPS, for location sensing, compass, gyroscope, and accelerometer for movement sensing, cameras, and many other sensing functionality, mobile devices can provide real-time contextual information about users and their environments [4], [30], [51]. This development provides unprecedented opportunities to develop solutions that can assist mobile users to overcome the challenges of mobile environment involving ubiquitous consumption of multimedia content. Context-aware personalized media recommendations will afford mobile users the ubiquitous access to multimedia items they desire in shorter time. Additionally, it will also offer mobile users the ability to navigate straight to the relevant items based on their contextual situations, thereby minimizing the volume of irrelevant items that always pop up when they search for items of interest.

Nevertheless, in spite of these promises, existing mobile media recommendation systems use content-based filtering or collaborative filtering to deliver explicitly requested media items to mobile users, based on time contexts and locations [50], [53]. But, proactive and sensor based mobile recommendation techniques will revolutionize anytime, anywhere multimedia consumption, using mobile users' contextual information, such as location, activities, environment information, etc., to deliver, with minimal intervention of the users, relevant multimedia recommendations [4]. Context-aware media recommendation is an exciting new body of research with peculiar challenges that require adequate attention because of its importance in the field of mobile technology. In the next sections, the background of the thesis, especially in recommendation systems, context-aware computing, user profiling, and media presentation adaptation, will be discussed.

2.2 Multimedia recommendations: traditional approaches

To address the so-called "*information overload problem*", in which information consumers cannot find what they want due to the massive volume of myriad available alternatives, personalization techniques have been extensively explored to deliver information to users according to their personal interests [23], [27]. Recommendation systems (RSs), one of the fields of personalization, have been used effectively to address the information overload problem, especially in desktop systems. As depicted in Figure 2.1, traditional recommendation systems take information about items and users and process this information to suggest items of interest to target users. In their seminar paper, Resnick and Varian [54] define recommender systems as systems that provide recommendations to people by aggregating and directing them to the appropriate recipients

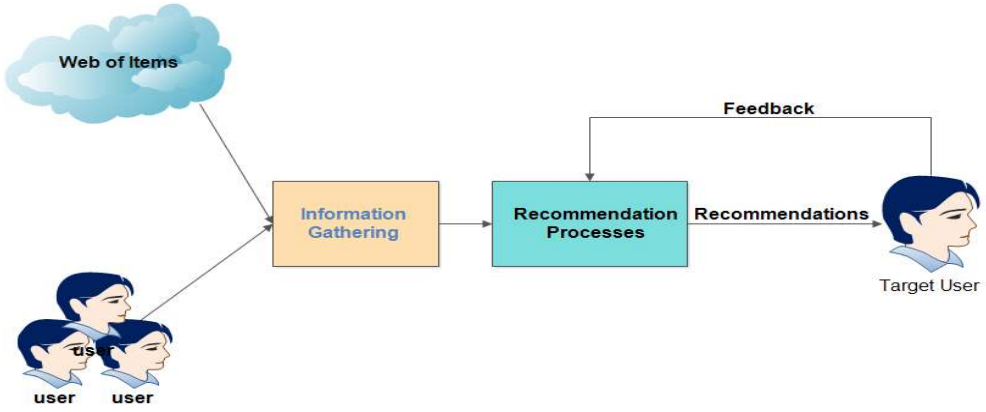


Figure 2.1- A typical traditional recommendation process

Later, Burke [28]-[29] defined recommendation system as "*any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting and useful objects in a large space of possible options".*
This definition is much broader than the one given by Resnick, where he essentially defined a recommendation system as a collaborative system. In [23], Adomavicius and Tuzhilin gave a formal and widely cited definition of RSs, covering all types of traditional RSs thus: "*… Let C be the set of all users and let S be the set of all possible items that can be recommended. Let u be a utility function that measures the usefulness of item s to user c, that is, u: C x S →R, where R is a totally ordered set (for example, nonnegative integers or real numbers within a certain range). Then, for each user c ∈ C, we want to choose such item s' ∈ S that maximizes the user's utility*. Ricci et al. [3] also defined RSs "*as software tools and techniques, providing suggestions for items to be of use to a user. The suggestions relate to various decision-making processes such as what items to buy, what music to listen to or what online news to read"*.
These definitions have been the most comprehensive of all definitions of RSs found in the literature. From the above definitions, it is clear that the main goal of these systems is to reduce the user's search efforts through a corpus of content by listing and presenting those items with highest utility or relevance to the user [29]. Therefore, it is no doubt that RSs play important roles, especially in e-commerce and online multimedia systems. For example, these systems have been explored in e-commerce systems, such as Amazon.com, to help customers make purchasing decisions. They have also been used to help movie (e.g. Netflix), and music (e.g. Last. FM) lovers make informed decisions on the best and interesting movies or music they prefer to watch or listen to respectively. Large and highly rated online industrial multimedia service providers such as Yahoo, IMDB, YouTube, etc. have RSs deployed as part of their services.

Many other examples of these systems can be found in tourism [19], [30], targeted advertisement [31], health [32], transportation [32], music [34], news [9]-[10], commerce [35], [36], movies [37] and even online dating systems.
Traditionally, multimedia recommendation systems have focused on suggesting multimedia content to users, using information about the multimedia items the user has consumed in the past, designated as content-based filtering (CBF) or using information about those items that other like-minded users have consumed, designated as collaborative-based filtering (CF). Sometimes, a combination of these techniques is used to push relevant multimedia content to users, which is designated as hybrid recommendation system [10]-[11], [23], [28]-[29]. The core task of a traditional RS system is to predict the subjective rating a user would give to a media item he has not yet seen [30]. Normally, to realize this task, they explore either the ratings given by friends of the user to content consumed in the past to compute a rating for an item not yet consumed by the user, or they explore the content of the candidate media items and the preferences stored in the profile of the target user. Essentially, a recommendation system takes as input user information and the description of items, and processes the inputs to produce as outputs items likely to be of interest to the user. Additionally, to improve the relevance of future recommendations, a recommendation system learns the user's "*likeness*" (feedback) of the recommended items. These recommendation techniques have been developed to provide users with relevant multimedia content efficiently from a very large corpus of content. However, the main objective is the explicit multimedia content suggestion that is relevant to the preferences of the user in traditional environments. The recommendation is made when the user explicitly requests for the RSs assistance, and it is not expected that user's preferences would change with time and locations.

2.2.1 Content-based recommendation processes

Content-based recommendations, also known as content based filtering (CBF,) are recommendation techniques that provide items that are similar to those the user has preferred in the past [10]. CBFs rely on the user's profiles with historical records of items consumed in the past, and then matching the user profiles up with the profiles of those items that have not yet been consumed by the user to find most relevant items. The user profile also contains special attributes of the user that define his preferences for specific items. Consequently, CBF systems incorporate some basic techniques to represent the multimedia items, the user preferences/interests, and some strategies to compare the user

profile and the multimedia item profiles. Ricci [3] list these techniques as content analysis, user profiling, and filtering processes.

2.2.1.1 Content analysis

Generally, online-based multimedia content such as texts are not structured. Therefore, they are preprocessed in order to extract from them meaningful and structured information. This helps to structure the content of the multimedia items coming from the online sources, using feature extraction techniques. The product of this process is a representation of the content of the items, which have been sifted to match the candidate items. Examples are web pages or movie contents that are represented as keyword vectors [1], [10], [55]-[56]. The keywords or terms in the vector are associated with a real number called *tf*idf* (term frequency times inverse document frequency) that represents the importance or relevance of the keyword. Usually, as defined in [10], these real numbers are generated using equation 2.1.

$$w(t,d) = \frac{tf_{t,d} log\left(\frac{N}{df_t}\right)}{\Sigma_i \left(tf_{t_i,d}\right)^2 \log\left(\frac{N}{df_{t_i}}\right)^2} \tag{2.1}$$

The *tf*idf* weight, *w(t,d)*, of a term in a document *d* is a function of the frequency of *t* in the document *(tf_t,d)*, the number of documents that contain the term *(df_t)* and the number of documents in the collection *(N)*.

Where $tf_{t,d}$: is the term frequency, which measures how frequently a term *t* occurs in a document.

tf_t = *Number of times term t appears in document d divided by the total number of terms in the document*

$df_{t,d}$: *is the inverse document frequency, which measures the importance of term t in document d.*

$$idf_t = \log_e \left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t}\right) \tag{2.2}$$

However, one important factor to note is that *w* in the above equation does not capture the situations in which those multimedia items are consumed, it only captures the term in the content as well as the importance of such term in the document.

2.2.1.2 User profiling

The user profiling process collects data that are representative of the user's preferences, generalizing the data to construct the user profile model [3]. The data generalization uses techniques, such as data mining, to infer the model of the user's preferences, based on data about multimedia items consumed or those not consumed by the user in the past [10]. Thus, the type of data that are contained in the user profile can be defined as primarily containing the preferences or interests of the user. User preferences are the descriptions of the types of multimedia items that a user has preferred in the past, which can be used to infer the items the users would like to consume in the future. User profiling usually employs some model that can predict, for any multimedia item, the likelihood that the user would consume such multimedia item. Generally, it retrieves *n* number of multimedia items, which are more likely to be preferred by the user based on the rating given by users to such items. Essentially, it contains information about the items that user has consumed or not in the past is stored (e.g. Information about the kind of movie, music, or news that the user has consumed such as the genre of such items). Another example of information contained in the user profile is the optional personal and demographic information such as age, sex, marital status, etc. can also be included in the user profiles. Section 2.4 reviews user profiling process in detail, and provides existing methods used to identify, represent, and process user profile information.

2.2.1.3 Filtering process

In the filtering process, the system exploits the user profile to match it up with candidate multimedia items to recommend the most relevant to users. The process ends with the generation of a binary or a continuous relevance judgment of the user, using some similarity measure such as Pearson or Euclidean distance as well as Cosine of the angles between vectors representing the user profile and those of the candidate multimedia items. Model based methods based on machine learning algorithms such as Bayesian model, Neural Network, Support Vector Machine, decision trees, etc., have also been explored in the filtering process [29]. Section 2.2.5 discusses some of these techniques.

2.2.1.4 User feedback

To understand the future needs of users and to personalize the recommendations, information about the satisfaction of the user concerning items consumed is solicited either explicitly or implicitly [10], [57]. This information is called feedback. The explicit feedback requires users to evaluate the relevance of recommendations by giving either positive or negative evaluations. Example of explicit feedback is the thumb up or thumb down on YouTube videos. The implicit feedback on the other hand does not require the user's involvement rather the system monitors and records user's interactions with the system or user's reactions to the recommendations, using the feedback information to improve the future recommendations. Example of implicit feedback is monitoring the number of times music tracks in iTunes has been played by a user or the time spent on a web page.

2.2.1.5 CBF's limitations

Though the content-based recommendation has been highly successful in news RSs, such as *netnews* recommender systems [9], it however suffers from the following limitations.

- **Over-specialization**: Generally, CBF recommends items with high scores against the user profile. This means that users are recommended items they have seen or those that are too similar to the ones they have seen before. For example, a user who scored a romance movie very high, and who has never watched an action movie, might never have action movies in her list of recommendations, should he/she prefer action movies. He/she would always have romance movies or those closely similar being recommended. It is, thus, necessary to incorporate mechanisms to allow CBF systems recommend items that users have never seen before. An example of such approach is the use of item attributes through diversification to determine the list of items that are dissimilar and yet relevant to the users as presented in [58].
- **New user problem**: This problem arises when a new user uses the system. In this situation, the system does not have adequate information about the consumption history of the user, thus suggesting new items to the user becomes a problem.
- **Limited content analysis**: Content-based recommender system requires the representation of item content [23]. This is a common technique in information retrieval called feature extractions. However, in some domains, this is difficult to realize. For example, it is difficult to extract automatically the features of multimedia items such as video, music, and news. An alternative way is to associate the features manually to items. This is a difficult and expensive process considering the necessity of employing dedicated computing resources to extract features from millions or possibly billions of items. The latter may not always be available and additionally, each annotation that has been manually inserted is highly subjective and is dependent on the individuals who insert them.

2.2.2 Collaborative-based recommendation processes

A Collaborative-based recommendation, also known as collaborative filtering (CF), provides recommendations to a user based on the opinions of other like-minded users [11], [29], [54]. In [23], Adomavicius and Tuzhilin formally defined CF as the "utility" $u\,(c, s)$ of items for user c, is estimated based on the utilities $u\,(c_j, s)$ assigned to item s by those users $c_j \in C$ who are similar to user c. This definition is based on the assumption that finding items to recommend

to a user, other users must have seen and evaluated the items. For example, in a movie recommendation application, to recommend movies to a user c, the system must first find the so-called "friends" or "neighbors" of user c, who have similar movie preferences. The movies recommended to user *c* are those most preferred by his friends. In the literature, CF approaches are categorized into two major classes: the memory based and model based collaborative recommendations [23].

2.2.2.1 Memory based collaborative processes

Memory based collaborative filtering (MBCF) uses the entire collection or samples of user-item dataset to generate a prediction for an unseen item. Every user belongs to a group that has a similar preference. Therefore, the primary task of a memory based CF is to determine this group of people with similar preferences to the target user c. The most popular technique to execute memory based CF is the so-called neighborhood approach, and it is realized in the following steps.

(i) First, it calculates the similarity or weight $w_{c,s}$ (using distance correlation, such as Cosine, Pearson correlation, etc.) to determine the closeness between two users or items, c and s.

(ii) It then generates a prediction for the target user, by taking the weighted average of all ratings of the user or item on certain items.

(iii) To produce a *top-N* recommendation, the system finds the *k* most similar users or items. It then aggregates the neighbors to get the top-*N* most relevant items as the recommendations.

MBCF can be broadly categorized into two types: Item and user based MBCF [59].

(a) User based CF

Most CF based algorithms can be classified as user-based recommendation algorithms because the predictions for users are based on the ratings of items given by users who are similar to the target users [23]. For example, given a target user u_1 and a set of users, say, $U = (u_2, u_3, u_4, \ldots, u_n)$, user based CF will generate predictions for an item *i* by analyzing ratings for *i* from users (U) who are similar to u_1. This process captures how traditional word-of-mouth recommendation works and it can detect complex patterns if used with adequate user ratings. In practice, however, user based CF suffers from some major challenges. First, the rating data are always sparse, and pair of users with few common ratings is prone to skewed correlations. Second, it fails to incorporate agreement about an item in the item corpus as a whole. For example, if two users agree on universally accepted items, the importance of such agreement is less significant than those items that are less generally accepted. Third, finding the user's neighbors, especially in systems with, say, millions of users, is computationally expensive because it will require comparisons against all other users. This explains why commercial systems such as amazon.com, with millions of users cannot scale with user-based CF. However, with a sizeable number of users; this recommendation approach has proved to be much more efficient [60].

(b) Item based CF

Because of the weaknesses of the user-based CF, item-based CF was proposed to address those problems. Whereas user based CF generates predictions for target users based on the ratings given to items by similar users, the item based CF is the transpose of user based CF, where ratings of similar items are used to generate predictions for items to be recommended to the target user. There are compelling evidences in the literature that this approach produces more quality recommendations than user based CF, but with a large (millions or billions) corpus of items, the size of the item based recommendations can shoot up to as large as the square of the number of items [61]. This approach is also computationally expensive with large volume of items, and scalability issue remains a serious concern. However, there have been successful attempts to address this problem. Sarwar et al. [61] developed a solution, which reduces

the size of the items by using *topN*-correlated items with *k co-ratings*. Other methods based on matrix factorization techniques have been developed, and their details can be found in [62].

In summary, MBCF based recommendation approaches are generally more accurate and very useful, especially for real-time recommendation processes because they do not require data training process, which usually is time consuming and computationally expensive [23], [64]. Even though, they are susceptible to the cold-start problem [23], MBCF based recommendation processes in mobile environments will benefit from their ability to incorporate real-time data as well as better recommendation accuracy they offer. Additionally, its cold-start problem can be addressed using contextual information rather than using rating information to compute recommendations, as proposed in this thesis.

2.2.2.2 Model based collaborative process

This technique uses a collection of user ratings of items previously consumed to learn a model, which is then used to predict ratings for new items [23]. It uses machine-learning techniques, such as clustering and classification algorithms to learn the recommendation model. Model based methods such as the Bayesian model, singular value decomposition (SVD) model, Artificial Neural Network (ANN), rule based classifiers, etc. have been explored to build model-based recommendation systems [63].

2.2.2.3 General limitations of CF systems

- **New Item problem**: CF requires an item to have been sufficiently rated by users before it can be recommended accurately to a user who has not seen it before. In fact, a new item that has not been rated adequately cannot be considered for recommendation. Nevertheless, this type of problem can be solved using content-based recommendation since it only requires the content of the new item and information about the user to consider it for recommendation. Please note that CF also suffer from new user problem just like the CBF systems.
- **Sparsity problem**: When the number of ratings to predict is more than the number of ratings already obtained, recommender system suffers from sparsity problem. This problem can be addressed, using hybrid recommendation techniques. However, some existing work used matrix factorization approach such as SVD model to provide lower ranks of the original sparse matrix [23].
- **Intrusiveness**: Users have to supply information explicitly to the system (e.g. their tastes or preferences). In practice, it has been proven that many users are reluctant to provide this information, especially in a mobile environment.

2.2.3 Hybrid recommendation processes

Although collaborative and content-based filtering techniques have both been successful in research and in practice, they both present some weaknesses as discussed in the last section. To address those weaknesses, the hybrid recommendation technique has been proposed, combining two or more recommendation techniques, mostly collaborative and content-based recommendations. The main idea behind the hybrid recommendation according to Mark Setten [65] is *that a combination of two or more recommendation algorithms can provide more relevant recommendations than a single algorithm, and that the disadvantages of one them can be overcome by the other algorithm*. Therefore, incorporating content-based filtering into collaborative recommendation, for example, has been proven to improve recommendation system quality [66]. Besides, the sparsity problem, which is very prevalent in collaborative recommendations, can be addressed with additional content information provided by the content-based technique [28]. A good example and one of the earliest hybrid recommendation systems is the Fab system [66]. Fab is a content-based collaborative recommendation system, which recommends web pages to online users. The hybridization process is realized in two stages: collection of items to form a manageable database or index of items and the subsequent selection of items from this database for a particular user. Many other hybrid systems have been developed, details of these systems and methods can be found in [28]. Nevertheless, most of these existing hybrid

recommendation systems differ in the way they combine recommendation algorithms. In the next sections, seven different approaches to achieving hybridizations as evaluated by Burke [67] are discussed.

2.2.3.1 Weighted hybridization

The weighted hybridization uses weighting to combine the prediction of two or more pure recommendation techniques. The simplest way to combine the central prediction is to calculate the average of the predictions, and using the result to provide recommendations. Its main weakness is that it could sometimes produce predictions that are less accurate than the pure recommendation technique. This problem arises from the underline function used by the weighting process. For example, if statistical central tendency functions such as media, mean, mode, etc. are used, then the final ratings could be situated between the lowest and highest initial predictions. If the initial ratings are situated to the left or to the right, then the final predicted rating could be less accurate. To overcome this problem, more advanced non-linear techniques have been used. For example, machine learning techniques such as Bayesian Network, Neural Network, etc., can be used to combine the predictions of two or more pure techniques.

A good example in this category is presented in [18], where content-based recommendation is used to measure the similarities between media items and user preferences, and using a Bayesian classifier to determine the probability that the media item belongs to specific situational contexts via a weighted linear combination of the scores to obtain a final prediction score.

2.2.3.2 Switching based hybridization

In switching, the hybrid recommendation systems switch between all the pure techniques depending on some criteria. For example, if one technique is not capable of producing accurate predictions, then it uses the other technique. To use this method, the hybrid system must have adequate understanding of the constituent recommendation techniques, for instance, they should be fully aware of when they are able to produce accurate or inaccurate recommendation predictions. Incorporating this knowledge into the hybrid system is not a trivial process. However, the benefit of such systems is that it is sensitive to the strengths of its constituent recommendation techniques. Many systems use this approach by combining both content-based and collaborative recommendations. In [68], the system presented uses Bayesian classifier in an offline stage to generate recommendations. However, if the predictions computed by the Bayesian classifier are less accurate, it uses item based collaborative recommendation in the online mode to generate recommendations.

2.2.3.3 Mixed hybridization

Mixed hybrid recommendation systems, unlike the switching-based hybrid systems, do not combine the pure techniques to realize hybrid process. Rather, the prediction produced by each pure technique is used to generate the recommendations, which are then presented to the user. Therefore, each item presented to the user has multiple predicted ratings associated with it. One advantage of this approach is that the system implementing it does not have to have complete knowledge of its constituent pure methods. The *TasteWeights* system in [69] uses first, a content-based algorithm to generate Wikipedia content that relates to a given topic. It then uses collaborative filtering to generate Facebook contents. The recommendations from each method are ranked, and the first top *n* items from each method are combined, and presented to users. This approach addresses the new item problem because the content-based component can be relied upon to provide recommendations based on the description of the item content, even if adequate number of users has not rated it. However, because both CF and CBF suffer from new user problem, this mixed hybrid system still suffers from new user problem.

2.2.3.4 Cascaded hybridization

In this approach, the first recommendation method (called high-priority technique) generates an item prediction in a coarse form, which is then refined by the second method (called low-priority method) in a staged process. Cascading hybrid recommendation techniques allows the system to avoid employing the second low-priority technique on items that are well classified by the first recommendation technique or those that are sufficiently poorly rated that they might

never be recommended. By its nature, the cascading method can tolerate noise in the low-priority recommender process since the ratings given by the high-priority can only be refined and not discarded [28]. For example, the system presented in [68] uses a cascade of content based filtering to process music genre to generate music content more likely to be preferred by a user, and then uses the personality diagnosis filtering to generate the final music recommendations.

2.2.3.5 Feature combination hybridization

In feature combination, features of one data source are injected into the recommendation algorithm to process data from another source. For example, features from collaborative information are treated as additional features data of the items in a content-based filtering to generate recommendations. This means there is an actual recommendation process and another "virtual" recommendation process. Those features that would have been processed by the virtual process are instead used as part of the input to the actual recommendation process. Therefore, in the real sense, feature combination is not a hybrid recommendation because only one recommendation process is involved. For example, in [70], an association rule mining technique is employed in a discussion group recommender system, to discover like-minded users. It then uses content-based technique to recommend related posts to the users. This is a feature combination hybrid recommendation that adapts logics from another recommendation rather than implementing the two recommendations: What makes it a hybrid recommendation is the knowledge sources involved in the process.

2.2.3.6 Feature augmentation based hybridization

This is a hybrid strategy that is similar to feature combination. However, rather than using features drawn from the contributing recommendation methods, the feature augmentation generates a new set of features for each item, using the logic of the contributing recommendation method. The contributing recommender normally intercepts features headed for the main recommender and augments it with some computation. Its primary goal is to strengthen the existing, well-developed recommendation process. Because the augmentation can be done offline, the recommendation process is performed faster, which makes it more attractive than other hybrid approaches, including feature combination. The most notable application of this method uses content-based algorithm to generate ratings for unrated items, and then using collaborative recommendation to generate the final recommendations. For example, Wang et al. [46] use user profile in a content based recommendation to generate ratings for news items, the generated ratings are then used in a collaborative process to generate the final predictions for each news item for the target user.

2.2.3.7 Meta level based hybridization

The metal level hybrid approach uses a learned model by one recommender as input to another recommender to generate recommendations. Unlike the feature augmentation, the contributing recommender completely replaces the original knowledge source of the main recommender with the learned model, which the main recommender now uses in its computations. The main recommender does not use raw data from the user profile. For example, it first builds a model of the user preferences, which is then used in the collaborative process to provide recommendations. In [71], Campos et al. use a Bayesian network-based user profile model in a content-based approach, and uses the model in the collaborative process to produce recommendations.

2.2.3.8 Weaknesses of traditional hybrid recommendation processes

Hybrid recommendation systems, using a mixture of content-based and collaborative-based recommendation algorithms have been explored extensively in research and industry domains [28]. However, because both content based and collaborative recommendation systems suffer from new user problem, a hybrid recommendation system exploring them also inherits the new user problem; Table 2.1 is a summary of the strengths and weaknesses of these traditional approaches. Additionally, most hybrid recommendation hybridization approaches are static because they do not reflect changes in user preferences at runtime [72]. Consequently, they cannot provide optimal recommendation quality in some domains where user preferences change frequently. For example, in mobile environments, because users are always on the move with their devices, their interests, preferences and desires are transient and are subject

to change. A good recommendation system should reflect these changes at runtime when making predictions. The traditional hybrid recommendation systems do not have these properties.

Table 2.1 - Weaknesses of CBF, CF and hybrid recommenders

Technique	New User	New Item	Sparsity	Overspecialization	Limited Content Analysis
CBF	√	√	×	√	√
CF	√	√	√	×	×
Hybrid	√	√	×	×	×

2.2.4 Context-aware recommendations

Conventional recommendation techniques as discussed in the preceding sections recommend items to users with an assumption that the ratings given by users to previously consumed items do not change. Additionally, users have to request media items explicitly to initiate the recommendation process. However, in practice, user's preferences or the preferred content types change as they move from one place to another. The recommendation systems, therefore, should provide relevant media items that are suitable for users according to their changing preferences and situations. Adomavicius et al. [15] are the first set of researchers to argue in favor of incorporating contextual information in recommender systems. They proposed that it is important to consider context information when providing recommendations because contextual information intuitively characterizes the consumption situations of users. For example, the content that a typical user would prefer at home would be different, in most cases, from those she would prefer at work. For instance, a user would not watch movies at work, but would probably prefer to read latest or breaking news or listen to music while performing work related activities. This condition has been largely ignored by the traditional recommendation systems. Thus, these existing recommender systems do not take into consideration contextual information such as location, time, activity, etc. in the recommendation process [15], [20], [73]. However, in recent years, researchers have begun to incorporate contextual information into recommender systems, popularly known as context-aware recommender systems (CARS) [15], [73]. Adomavicius [15] et al. proposed multidimensional methods to incorporate contextual information into recommendation systems in e-commerce domain, and many researchers have proposed different contextual recommendation systems to address the weaknesses of the traditional recommendation system techniques [4]. Figure 2.2 (a&b) illustrates the difference between traditional recommendation systems and their context-aware counterparts. The traditional recommendation is a two dimensional process with user and item domains, whereas, context-aware recommendation is a multidimensional process with at least three dimensions namely, users, items, and context dimensions.

In this thesis, the definition of the traditional recommendation given in section 2.1 is modified as follows to define the context-aware recommendations.

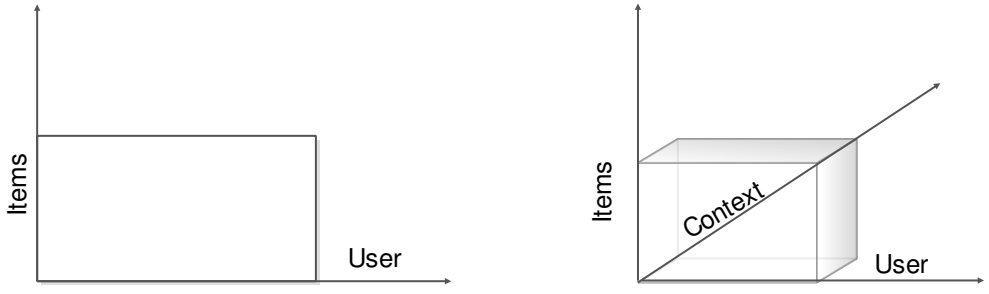


Figure 2.2-(a) Traditional recommendations (b) Context-aware recommendations

Let C be the set of all users (with mobile devices) and let S be the set of all possible items that can be recommended. Let u be a utility function that measures the usefulness or relevance of item i to mobile user c, in context l that is, u:

C x S x l →R, where R is a totally ordered set (for example, nonnegative integers or real numbers within a certain range). Then, for each user c ∈ C, we want to choose such item s' ∈ S that maximizes the user's utility in context l ∈ L. Context *l* in this definition specifies the information that characterizes the mobile user, her consumption environment and preferences, and her devices. The dynamic preferences of mobile users based on location, time, activity or other environmental information introduce other dimensions to the recommendation systems, and thus new challenges.

Context-aware recommendation systems have been proposed in tourism domain, in music domain, in the movie domain, in the mobile TV domain, in the news domain, etc. [4]. For example, mPERSONA [43], one of the earliest mobile recommendation systems, presents flexible content recommendations for wireless users, taking into consideration user's mobility, her local environment, and her device profile. Another interesting work is uMender [34], which groups users in similar contextual conditions, and recommends relevant music items that match their listening interests and the current contexts. There are several other research efforts, such as [26], which use context information to provide advertisements to mobile phone users. The contextual approach to recommendation and personalization has the ability to address the weaknesses, especially new user problem, of the traditional approaches. However, the major challenge for developing practical context-aware recommendation system remains that of acquiring contextual information dynamically. Present context-aware recommender systems require an explicit procedure for context information acquisition [4]. Explicit acquisition of contextual information will lead to additional burden on the part of the busy users, and this will discourage users from appreciating the benefits of context-aware recommendations. Proactive, mobile phone sensor-based context-aware recommendation systems that can detect user's contexts in real time, with little or no user intervention, will revolutionize context-aware recommendations.
In section 2.3, the current chapter deals more on the concepts and definitions of context including how it can be obtained and applied in practical applications.

2.2.5 Recommendation system algorithms

In the literature, various techniques and algorithms have been employed to address recommendation problems. These methods generally use classification algorithms. Classification is a process that maps the feature space and the label space of a data set [63]. The features are the characteristics of the elements to be classified and the labels represent the classes [12], [61], [63]. This section reviews some of these methods, especially the most common ones, and those similar to the algorithms explored in this thesis.

2.2.5.1 k-Nearest Neighbor (kNN) recommendations

kNN belongs to a class of classification algorithms referred to generally as the instance-based or lazy learners [63]. They work by storing training records, and using them to predict labels of unseen cases. kNN memorizes the entire training set and classifies only the attributes of a new record to match exactly one of the training examples. It finds the k closest points, given a point to be classified, from the training records, by assigning the class labels according to the class labels of its nearest neighbors. The basic principle of kNN is that if a record falls in a particular neighborhood, where a class label is predominant, it is because the record most likely belongs to that class. For instance, given a query *a* (which could be the profile of a user or her rating record) for which we want to recommend a class *c* of items and a training set $X = \{\{x_1, c_1\} \ldots \{x_n\}\}$, where x_j is the j_{th} element and c_j is its corresponding class label. Thus, the k-nearest neighbor algorithm finds a subset $Y = \{\{y_1, c_1\} \ldots \{y_k\}\}$, such that $Y \in X$ *and* $\sum_1^k d(a, y_k)$ is minimal, *Y* contains the *k* points in *X*, which are closest to the query *a*. The class label *c* for query *a* is given as $f(\{c_1, \ldots, c_k\})$ [63].
kNN based recommendation systems are very popular, especially in collaborative based recommendation systems because intuitively, kNN is related to the CF idea of finding like-minded users or similar items (neighbors). Most existing collaborative filtering based recommendation systems have been implemented based on kNN algorithms [11]. A very good example is the work presented by Chen [14], where items consumed by like-minded users in similar contexts are used to predict new item to a user in the present context. Other examples of kNN, especially traditional recommendation systems, can be found in [11], [74]. The main challenge, however, of kNN-based recommendation systems is that of choosing an appropriate value of k. If the value of k is too large, then the neighborhood might include

many points from other classes. If the value of k is too small, the system will be too sensitive to noise. However, through experimentation, this problem can be resolved. kNN-based recommendation systems come with a very important benefit. Because it is a lazy learner, a kNN-based system does not require learning and maintaining a model. In this case, it can adapt very rapidly to changes in the user/item-rating matrix. Though very simple and intuitive, kNN based systems are very amenable to improvement and always give accurate recommendations. However, kNN-based recommendation systems suffer from scalability problem because it loads the entire data set into memory during the classification process. With a large number of items or users, scalability becomes its bottleneck. This problem can be addressed using matrix dimensionality reduction approaches such as principal components analysis (PCA) and singular value decomposition (SVD) [63]. For example, Barragáns-Martínez et al. [75] presented quevo.tv, a TV program hybrid recommendation system that uses SVD to address the scalability problem to achieve good performance. In context-aware recommendation domain, Gupta et al. [88] addressed this problem using SVD to achieve efficiency.

2.2.5.2 Rule based recommendations

Rule based systems classify items based on a collection of "if… then…" rules. The antecedents or conditions of the rules are expressions made up of attributes and conjunctions. The rule consequent is a positive or a negative classification. A rule *r* covers a given instance $\varkappa$ of the attributes if the instance satisfies its antecedents, whereas the accuracy of the system defines a fraction of the records that satisfies both the antecedent and the consequent. In addition, a rule-based system is mutually exclusive if its rules are independent of each other. Because this approach uses symbolic attribute to operate on data without any transformation, these systems are very expressive. Rule based systems are easy to generate and can classify new instance more accurately. However, building rule based recommendation systems is a very difficult and complex process because it requires an explicit knowledge representation of the recommendation decision-making process. The rule-based classifiers have been used to complement RS by injecting partial domain knowledge [63]. The work in [76] is a good example of rule-based recommendation systems, which applies rule-based classifier to complement a collaborative recommendation process to help teachers maintain, and to continually improve e-learning courses. Other examples of rule based recommender systems are meta-mender [97] and Comtella-D [77]. Meta-mender uses rules to provide effective recommendation to learners while Comtella-D explores a rule-based model to suggest relevant discussion fora to online users.

2.2.5.3 SVM-based recommendations

Support Vector Machine (SVM) is a classification algorithm that finds a linear hyperplane, which separates data in such a way that the margin between the data is maximized. For example, given some training data D, a set of n points of the form:

$$D = \{(x_i, y_i) | x_i \in R^p, y_i \in \{1, -1\}\}_{i=1}^{n} \quad 2.1$$

Where y_i is either 1 or -1, indicating the class to which the point x_i belongs. Each x_i is a p-dimensional real vector. The goal is to find the maximum margin hyperplane that divides the points having y_i = 1 from those having y_i = -1. The linear separation between the two classes is realized by defining a function:

$$f(x) = W.X + b = 0 \quad 2.2$$

Where ". "denotes dot product and W the normal vector to the hyperplane**.** Another function is then defined to classify items belonging to a given class +1 or -1 as long as they are separated by some minimum distance from the class separation function (2.3):

$$f(x) = \begin{cases} +1 \; if \; W.X + b \geq 1 \\ -1 \; if \; W.X + b \leq 1 \end{cases} \quad 2.3$$

To maximize the margin $\frac{2}{\|w\|^2}$ between two classes, the merging function is an inverted object to the function $f(x)$. In terms of performance and efficiency, SVM based systems have become very popular. Oku et al. [138] extended SVM based classification process with context information and then applied it to a CF based restaurant recommendation to suggest relevant restaurants to users based on time, their partner's information, schedules, weather, temperatures etc.

2.2.5.4 Bayesian based recommendations

Bayesian classifier has been extensively explored for model-based recommendation processes. This algorithm considers all features and classes of a learning problem as randomly continuous or discrete variables. A Bayesian classifier uses conditional probability and Bayes Theorem to maximize the posterior probability of any class of a given data for classification. This set of classifiers uses probability to represent uncertainty about relationships learned from data. Additionally, its core concept of priors is an important representation of expectations or prior knowledge about what true relationship might be. The probability of a model given the data (posterior) is proportional to the product of the likelihood, times the prior probability (or prior). The likelihood includes the effect of the data, whereas the prior specifies the belief in the model before the data were observed. Bayesian classifiers consider each attribute and class label (continuous or discrete) as random variables. Formally, given a record with *N* attributes (A_1, A_2, A_3,…, A_N), the goal of the Bayesian classifier is to predict class C_k by finding the value of C_k that maximizes the posterior probability of the class given the data $P(C_k | A_1, A_2, A_3, \ldots, A_N)$ by applying Bayes theorem $P(C_k | A_1, A_2, A_3, \ldots, A_N) \alpha P(A_1, A_2, A_3, \ldots, A_N | C_k) P(C_k)$. Naïve Bayes is a very popular example of Bayes classifier, which assumes the probabilistic independence when estimating $P(A_1, A_2, A_3, \ldots, A_N | C_k)$ i.e. the presence or absence of a particular attribute. The main advantage of the classifier is that it is robust at isolating noise points and irrelevant attributes. They handle missing values robustly by ignoring such instance, during probability estimation. However, its main weakness is its independence assumption, which does not always hold for some attributes as they might be correlated. To address this issue, Bayesian Belief Network (BBN) has been explored [10].

BBN is an example of Bayes classifier that uses an acyclic graph to encode the dependence between attributes via a probability table that associates each node to its immediate parents. BBN robustly handles incomplete data and it is particularly good at addressing models over fitting. Bayesian classifier has been largely used in the traditional content based and collaborative recommendation systems [4]. For example, Miyahara and Pazzani [78] used Bayesian classifier in a collaborative recommendation process based on two classes, namely negative and positive ratings (respectively for "I like it" and "I don't like it") to recommend movies to users. In content-based recommendation, Meteren and Someren [55] present PRES, a Bayesian based content recommendation, which recommends small articles about home improvements to users. Bayesian classifier has also been used in context-aware recommendation process. In [79], Moon-Hee Park et al. proposed a context-aware recommendation system, which uses user-preferences, restaurant class, user's mood, restaurant price, and location as inputs to a Bayesian network model to recommend relevant and preferred restaurants to mobile device users. In [80], a recommendation system that extends Bayesian classifier with contextual information to suggest user-generated content to mobile devices is presented.

2.2.5.5 Neural network based recommendations

Artificial Neural Network (ANN) is a classification algorithm, which is an assembly of interconnected nodes and weighted links, inspired by the architecture of biological brains. The nodes in an ANN are called neurons, analogous to the biological neurons. These simple, functional units are composed into networks, having the capability to learn a classification problem after they have been trained with sufficient data. The simplest model of ANN is the perceptron model, which is a linear classifier that has a simple and efficient learning algorithm. In addition to the simple threshold function used in the perceptron model, there are other common choices for activation function such as sigmoid, tanh, and step function.

An ANN generally has a number of layers. Layers in an ANN are classified into three types: input, hidden, and output layers. The input layer responds to data that are fed into the network. The hidden units respond to the weighted output

from the input, and generate the final output from the input of the network. Using neurons as atomic units, many possible architectures of ANNs can be designed for a network. However, the most common architecture is the feed forward ANN, in which signals are strictly propagated in one direction i.e. from the input to the output. The main advantage of ANN, depending on its chosen activation function, is that they can perform non-linear classification tasks, and that the classification tasks are performed in parallel. Therefore, they function efficiently even if parts of the network fail. However, it is more difficult to come up with an ideal network topology for any given problem [90], which is ANN's main weakness. In addition, ANN works well when enough data are used to train the model. This could be its main weakness as a recommendation algorithm because of the data sparsity problem. ANN models have been applied in recommendation systems. Very good example is the work presented by Krstic and Bjelica [81], which uses a single-layered feed forward ANN as a classification model in a content-based recommendation system to recommend TV programs to users. The model is fed with context information such as time and the genres of the TV programs. Other work presented by Christakou and Stafylopatis [82] uses an ANN model in a hybrid recommendation process for movie recommendations.

2.2.5.6 Case-based reasoning (CBR) recommendations

Case based reasoning or CBR is a classification method that solves a new problem, remembering a previous similar situation, and by reusing information and knowledge in that situation [83]. It relies on the general knowledge of the problem domain or making association along generalized relationships between problem description and conclusion. It uses the specific knowledge of previously experienced, concrete problem situation (cases) to solve a newly identified problem. The new problem is solved by finding a similar past case and reusing it in the new problem situation. One unique and important feature of CBR is its ability for incremental and sustained learning from a new experience, which is retained each time a problem is solved, making it immediately available for future problems [83]. Another important advantage of CBR is that it can be used as a methodology and not as a concrete implementation of technology to solve a problem [84]. Considering the definition of CBR, which is about solving new problems by adapting solutions to old problems, it is not clear what specific algorithms or solution should be used. In fact, CBR generally follows four important cyclic methods to solve problems, which are:

(1) Retrieve similar cases to the problem descriptions.

(2) Reuse the solution suggested by the similar cases.

(3) Revise or adapt that solution to fit the new problem if necessary.

(4) Retain the new solution once it is confirmed or validated.

Therefore, how to address the problems or what technologies to use are left open to problem solvers. In this regard, CBR methodology has been explored in many problem domains. For example, in [85], CBR was used to recommend music that other users have listened to in similar contexts. In [86] Bouidghaghen et al. used CBR methodology in a context-aware user interest search system to select an appropriate user profile to re-rank search results.
In the proposed context-aware media recommendation system, CBR is adopted as a methodology for learning the contextual user profile, using it to retrieve past user's contextual preferences stored in the user profile model, or contextual preferences of other users in the knowledge base, based on the currently recognized user's contextual information. To accomplish this objective, four major processes corresponding to the processes, making up the CBR cycles have been developed, which are: *retrieve contextual preferences*, *reuse contextual preferences*, *revise contextual preferences*, and *retain and update contextual preferences*. The CBR based preference model uses k-nearest neighbor algorithm for retrieving similar cases. In order to compare the existing k-contextual preferences and current contextual preferences, CBR naturally supports the similarity function to compute the distance between these preferences.

2.2.5.7 Matrix factorization based recommendations

Often, the computational complexity of recommendation systems grows with the number of users and items. Generally, existing RSs commonly have sparse data set with a limited number of features per item. Data set with high dimensional feature space results in the so-called *Curse of Dimensionality* [90], in which recommendation systems suffer from scalability problem and consequently not able to provide quality recommendations. To overcome these challenges, the recommendation system community has explored approaches based on matrix factorization (MF). The MF characterizes both items and users by vectors of factors inferred from the rating patterns. High correspondence between an item and a user factor leads to a recommendation. Formally, MF works by assuming that ratings provided by users are represented in a sparse matrix $Y \in y^{nxm}$ where *n* and *m* are the numbers of users and items respectively. The observed values in Y are thus formed by the rating information provided by users. The problem to solve is that of matrix completion, i.e. mapping each user rating to an item in a user-item rating matrix. This problem has been addressed using techniques such as Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). SVD is a well-established method in information retrieval community [63], and has gained acceptance in RSs. The SVD can be used to factor matrix Y into three matrices thus:

$Y = U.\ S.\ V^T$, where U and V are two orthogonal matrixes of size *m* and *m* respectively. *m* is the rank of matrix Y, S is the diagonal matrix of size *m x m* having all singular values of matrix Y as its diagonal entries. All values of S are positive and are stored in decreasing order of magnitude. A very important property of SVD is that it provides the best rank approximation of the original matrix Y, in terms of Frobenius norm [87]. Therefore, it is possible to reduce he largest diagonal values to obtain a new matrix S_k, $k<m$. Thus, if matrices U and V are reduced accordingly, then reconstructed matrix $Y_k = U_k S_k V_k$ is the closest rank-k matrix of Y. Thus, Y_k minimizes the sum of the squares of the differences of the elements of Y and Y_k.

In the literature, SVD has been explored extensively in recommendation systems. Sarwar et al. [87] presented one of the earlier adoptions where they used SVD to capture latent relationships between users and items, allowing the computation of the predicted relevance of a given item by a user. Additionally, they used it to produce a low-dimensional representation of the original user-item matrix, and computed the neighborhood in the reduced matrix to generate a set of top-N items for recommendation. The context-aware recommendation system presented in [88] used a Bayesian classifier to provide relevant contextual information for users, using this information in an SVD to extract most important features corresponding to each item. One of the major advantages of SVD is that there are incremental algorithms to compute the approximated decomposition of the matrix, which allows the acceptance of new users or items without having to re-compute the model that has been built from the existing data. However, the main problem with SVD and MF in general is that in practice, it is difficult to re-compute the factorization every time the matrix is updated because of its computational complexity [63].

2.2.6 Techniques for evaluating traditional recommendation systems

The focus of traditional recommendation system researchers has centered on designing efficient and effective algorithms. Thus, developers wishing to incorporate recommendation services in their systems have a variety of algorithms to choose. However, the decision on which algorithm to select depends on the experimental evaluation of performance of individual algorithms compared with others using some evaluation metrics, borrowed from the field of information retrieval to provide ranking of candidate algorithms [57], [60], [89]. Literature review reveals that there is no standardized benchmark or framework for evaluating context-aware personalized recommendation system. Generally, recommendation systems researchers use existing metrics and experimental methods borrowed from other fields as noted above to evaluate the recommendation processes. In this section, experimental techniques and metrics adopted in the recommendation system domain are introduced with a view to understanding how the solution proposed by this thesis can be evaluated.

2.2.6.1 Experimental settings

The experimental methods found in the literature can be grouped into three major categories: offline, online and user study approaches [57]. First, we analyze these experimental methods and then present metrics used in recommendation system evaluations.

2.2.6.1.1 Offline experimental setting

The easiest experimental and most common setting for evaluating recommendation systems is the offline approach, also known as system-oriented evaluation [90], where existing dataset as well as procedure that simulates user model is used to evaluate the system without any direct user interactions. This experimental setup is used when accuracy (predictive or classification) of the system is chosen as the property to evaluate. Offline experimentation allows evaluations of properties of the system in different situations. For example, in context-aware recommendation systems, researchers usually evaluate the impact of context-awareness on the prediction accuracy of the system by first generating the prediction accuracy without contextual information, and then with context information, generate the prediction accuracy. The results are then compared to illustrate the improvement achieved due to incorporating contextual information in the recommendation processes. Essentially, based on its goal, evaluating recommendation systems based on offline approach requires that data used in the experimentation to be closely related to the ones expected in real life when the system is eventually deployed. This will help to have a good understanding of the performance of the system close to what obtains if it were operating in real life. Additionally, user's online behavior can be simulated where predictions or recommendations are made for the users, and users are allowed to accept or reject the predictions or recommendations in the simulation. This is done by collecting data from target users of the system. The data are then used to simulate how these target users would react when provided with recommendations. This experimental evaluation has an advantage over other two experimental settings in that it allows the simulation of target users in a laboratory setting without involving the more costly, expensive user studies, and online experimentations. That is, it is quicker and more economical than other experimental settings [57]. However, this method is subject to error, especially if experimental data collected are not close to those the system would encounter in real life situations.

2.2.6.1.2 User studies

In user studies, a group of users, usually called test users or subjects is recruited to evaluate the systems [57]. They are asked to perform recommendation requests using the system. The system then provides them with a set of recommendations. The test users are then asked to rate the provided recommendations explicitly or to quantify their overall satisfactions of the system. Prior to experimentation, data can be collected from these users via questionnaires. Using the information provided in the questionnaires, properties that are difficult to measure could be evaluated. For example, user's perceived quality of recommendations could be measured through this method. In many cases, they also ask other questions that are related to the qualitative performance of the recommendation system. User study can be executed in controlled laboratory experimental setting, in which users know that they are participating in evaluation of the system. This mode of user study is similar to system-oriented evaluation. On the other hand, the user study experiment can be conducted as a field-based experiment in which users are not aware they are participating in the evaluation. The advantage of user study-based evaluation over other experimental settings is that it allows researchers to test the behavior of the system while interacting it. Additionally, it allows us to understand the impact of user interactions or the behaviors of the users on the performance of the system.

2.2.6.3 Online experimental settings

In online or live experimental setting, performance is evaluated while users are using the recommendation system [89]. In cases where we want to evaluate not only the accuracies of the recommendation system but also other qualitative properties such as its ability to provide surprisingly interesting items to users, which is known as serendipity, live or online evaluation can be conducted[57], [39],[89],[91]. This experimental setting is similar to user studies and in fact is categorized in the same class as user studies or as user-oriented experimentation [90]. In most

cases, it is difficult to distinguish between user studies and online or live experimentation. However, online experimental evaluation allows measurement of the system's overall goal, possibly after extensive offline experimentations have been conducted. In literature, an example of online experimentations was developed by Hayes et.al [89] that simultaneously provides recommendations to a user using two recommendation systems. The goal was to determine which of the two systems provides better recommendations. In some other cases, both online and offline experimentation are executed together. For example, Yu et.al [18] evaluated their system combining both offline and online experimentations. Before recommendation request is made, in the offline phase, the system computes the predictions whereas, in the online phase, the system only determines the modality of the predicted item. The lesson from this approach is that, some aspect of the recommendation process can be conducted in the offline mode while other can be executed in the online mode. This allows executing costly processes in the offline mode and others in the online mode. One major weakness of online experimental evaluation, like the user studies, is that it is an extremely expensive experiment. To evaluate the overall goal and performance of a recommendation system based on this approach, it requires varying different properties of the system independently in order to understand the tradeoffs between these properties. According to Shani and Gunawardana [57], gaining a complete understanding of the system, using this expensive experimental procedure is very difficult. However, a simplified combination of these methods to limited scenarios has been explored to evaluate recommendation systems; a good example is CoMeR [18].

2.2.6.2 Evaluation metrics

As stated above, evaluating context-aware recommendation system is very expensive in terms of time, safety, trust, and resources [57]. This section analyses most common metrics used to evaluate the efficacies and quality of recommendations in literature. These metrics can be categorized broadly into two types, the predictive and classification accuracy metrics [60]. The classification metrics unlike the predictive metrics are popular for computing the relevance of recommended items because they do not directly predict ratings of items, but rather measure the ability of the system to provide relevant recommendations.

2.2.6.2.1 Classification accuracy

Classification accuracy metrics measure the frequencies with which a recommendation system makes correct or incorrect decisions about a recommended item or about whether the recommended items are relevant or not. *Precision*, *Recall*, and *F-Score* are the most popular metrics for evaluating classification accuracies of recommendation system. Precision can be defined as the proportions of the top recommended items that are relevant. It is the ratio of the number of relevant items selected from recommended items to the total number of recommended items. It represents the probability that a selected recommended item is relevant according to the preferences of the user. Recall on the other hand is the ratio of the selected relevant recommended items to the number of relevant items in the recommendation set. In other words, it represents the probability that the user would select a relevant item among the top n recommended items. F-Score is the harmonic mean or weighted average of precision and recall. It is a measure of the accuracy of the recommendation system in consideration of relevant and non-relevant items included in the recommendation set. In reality, precision and recall are inversely proportional and are both dependent on the number of recommended items returned to the user. When more items are returned, recall tends to increase whilst precision decreases and vice versa. These metrics are widely used in the information retrieval community because there are no standardized metrics for evaluating recommendation systems. It is important to note the difference in the usage of these metrics in both domains. In retrieval systems such as search engines, the ordering of the returned results is the most important factor. However, in most recommendation systems, what is important is the number of relevant items that are provided in the top n [92].

2.2.6.2.2 Predictive accuracy

Predictive accuracy metrics are also some of the most widely used recommendation system evaluation metrics. The metrics in this category are also borrowed from information retrieval and machine learning communities. The basic assumption for using these metrics is that a recommendation system that accurately predicts rating for an item will

most likely provide relevant items that will be preferred by the users. Usually, predictive accuracy metrics are used to evaluate systems that predict ratings (preference) for items those users have never seen, unlike the classification accuracy metrics that are used to measure the relevance of items provided by the recommendation systems. Predictive accuracy metrics measure the difference between rating provided by users to an item recommended to them and the ratings predicted by the system. Examples of this category of metrics are Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Normalized Mean Absolute Error (NMAE) explored by Meyer and Fessant to evaluate Reperio [74].

2.2.6.2.3 Qualitative metrics

Qualitative evaluation metrics measure the properties of recommendation systems other than predictive and classification properties. For example, measuring the "relevance" of recommendation to a user is not only about whether the classification or the prediction of a given item in the recommendation set is high or low but also about more of whether the user will like the consumed content. Examples of metrics in this category are novelty, serendipity, user satisfaction, etc. Serendipity for example, measures the ability of the recommendation system to suggest surprisingly interesting items to a user, i.e. it measures the un-obviousness in the recommendation of relevant items [57]. For example, a user usually likes items that are similar to items they have liked in the past or those items similar to those their friends like. However, what about items that would interest a user and yet is not similar to those liked by them in the past or similar to those of their friends? Serendipity measures the ability of a system to suggest such item [57]. Many researchers have used this metric to evaluate their system, e.g. Sarwar et al. [61].

2.2.7 Summary of traditional recommendation algorithms

Popular traditional recommendation techniques as discussed in this section have weaknesses, especially the so-called cold start problem. The hybrid technique, which was proposed to address these weaknesses, also comes with its weaknesses because the core of the problem lies with using explicit user ratings for generating recommendation. These techniques do not consider user's preferences in contexts such as time, locations, and activities. Therefore, these traditional recommendation techniques, which have proved highly successful in the traditional desktop environment, cannot be applied directly in mobile environments [30]. These techniques have to overcome the peculiar characteristics of the mobile environments. First, the limitation of mobile devices. Typically, mobile devices, even the most sophisticated ones, still have problems such as those posed by their screen sizes, and other problems associated with input information via the keyboard. These are not problems in the desktop environments. Second, wireless and mobile networks also have their own limitations. Mobile and wireless networks often are not reliable for consuming rich media items because of the fluctuation in the network resources such as bandwidth. Third, mobile users consume different media items depending on their present situations. The situation is characterized by their location, time, activity, environmental conditions, etc. In these situations, traditional recommendation methods are not realistic. The traditional recommendation approaches in their design do not consider this information as important factors that affect user's consumption preferences. They only consider the ratings given explicitly by a user or those given by the user's so-called neighbors.

In practice, however, users do not always give ratings of items they consume except the usage of such system is tied to the rating and even if they give, these ratings are always not adequate. The introduction of context-awareness into the recommendation process has been proposed as the future for delivering improved recommendation quality [15], [93]. Because contextual approaches use contextual information rather than relying mainly on ratings to generate recommendations, this new paradigm has the potentials to address the weaknesses of the traditional recommendation technique, especially the new user/new item problems. However, the present context-aware recommendation is faced with own challenges. The major challenge is that of acquisition of user's contexts in real time, and relating this contextual information with the preferences of the user to generate recommendations [52]. Additionally, these existing context-aware recommendation systems also, like the traditional recommendation systems, rely on static user ratings and explicit context information. The systems explicitly solicit for contextual information from users. In situations where users do not provide the contextual information or in those where they provide wrong contextual information,

this approach would affect negatively the recommendation quality, leading to the generation of irrelevant recommendations [4]. A truly context-aware recommendation requires the integration of dynamic context recognition into the recommendation system framework, which can provide equally high-level contextual information without user's intervention, one of the features that context-aware media recommendation (CAMR) incorporates to provide context-aware recommendations. The next section describes techniques/approaches that have been explored to incorporate contexts into RSs.

2.3 Modeling context information for personalization

This section defines contexts and describes how contexts can be incorporated into the domain of media item recommendations. It also discusses various techniques for context acquisition, inference, and representation.

2.3.1 Context definitions and concepts

Context-awareness or context-aware computing paradigm was first used by Schilit et al. applied to the field of ubiquitous computing, to describe the ability of computing devices to sense and react to the environment [94]. The ability of an application to adapt to locations, a collection of nearby people, devices, and to changes in this information over time is regarded as context awareness [94]. These systems are able to observe and react to things in their environment [13]. Schilit et al. [94] also states that context awareness concerns three aspects, *who you are, whom you are with, and with what resources*. Building upon this definition, Dey and Abowd in [95] gave a widely accepted and arguably the best general definition of context: *Context is any information that can be used to characterize the situation of an entity. The entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including users and applications themselves*. By this definition, context can be anything that defines the existence and the actions of anything! For example, in an ubiquitous computing environment, a typical computing device user is characterised by many information such as location(home, office, etc.), physical activities (driving, walking, running, sittting, standing, etc.), enviroment or location characteritics(noise level, weather, illumination, presence of other persons, etc.), time of day, or day of the week, personal information such as age, gender, etc.

The development in the ubiquitous and pervasive computing has engendered the ability of computing devices to sense, to examine, and to understand this contextual information in a way to serve the user better. Such information can be used to tailor information delivery or to autonomously assist users in their day-to-day existence. The use of context information, therefore, has become extremely important in the implementation of applications in pervasive and ubiquitous environments. Bettini et al. [96] identified two types of context information: based on their update rate, and based on their semantic level. The context information that is sensed in near real time are dynamic contexts. Because it does not change frequently, the context information, which is often supplied by the user, such as the information about their profession, is regarded as static context. However, the information held in the user profile, which Bettini et al. [96] categorized as static, essentially can be regarded as dynamic contexts because changing preferences due to changing locations or activities can not be regarded as static. At the semantic level, context information can be categorized into three types: low-level context information, high-level context information, and contextual situations, which are further discussed in the next sections.

2.3.1.1 Low-level contexts

These are raw data or events acquired directly from physical or software sensors with no further meaningful interpretations [8]. This context information is usually trivial and vulnerable to small changes and is usually uncertain in nature [96]. For example, events emitted by sensors such as accelerometers and gyroscope are mere raw data that do not make sense to humans and applications, or when evaluated *per sé* [38]. A change in their values might not be sufficient to take a decision on whether any reaction of the system is required. Instead, raw contextual data are often combined and subjected to reasoning techniques to derive new knowledge or context abstractions that model in a richer way the real-world situations.

2.3.1.2 High-level context information

To eliminate the limitations of low-level contexts, methods and models have been developed to infer meaning from the low-level sensor data. Such methods are usually called context reasoning and interpretation [8], [96]-[97]. The context reasoning and interpretation models take as input the raw sensor data, and generate as output high-level contextual information, also known as secondary context [98]. For example, in systems presented in [99]-[110], low-level data from sensors such as accelerometer are fed as input to a model to produce high-level contexts such as walking, running, jogging, driving, etc. This information can be used to learn user preferences for different categories of media content. For example, Mashita [111] developed system that can recognize users' physical contexts, such as walking, running, and standing to present mobile content using menu based presentation when standing, and a map based presentation when walking. However, most high-level context information does not make much sense. For example, it is difficult or even confusing inferring any meaning from "walking". "Walking to school" or walking to the office" will make more sense, and this high-level contextual situation can be explored by various applications to deliver services tailored or customized to the user's situations [38].

2.3.1.3 Contextual situation

A contextual situation or situational context is at higher semantic level than high-level context. A contextual situation is the product of semantic relationship between high-level contexts or between low-level and high-level contexts. It is a context information that can be expressed closer to the way humans express "conditions". In [98], Ye et al. defined situation context as follows: " *Situation can be defined by collecting relevant contexts, uncovering meaningful correlations between them, and labelling them with a descriptive name*". This type of contexts is meaningful and can be easily interpreted by humans and applications. For example, let us assume that a user u is located at home, and that she is sitting in the living room, and that the TV is switched on, we can conclude that *the user is watching TV*! The *user is watching TV* is a situational context. Also, *located at home*, *sitting in the living room*, and *TV on* are situation contexts. However, *sitting, home, TV, living room* (which can be obtained from the map of the "home") are secondary contexts or high-level contexts. These secondary or high-level contexts are usually obtained from low-level contexts, which are in turn directly obtained from sensors such as GPS, accelerometer, etc., as illustrated in Figure 2.3. In the next section, the chapters presents techniques for obtaining high-level context information from low-level sensor data.

2.3.2 Context recognition

Context awareness has been studied extensively in pervasive computing [73]. The acquisition, recognition, and the representations of contexts, which are the building blocks for effective use of context awareness are maturing. In this section, we discuss how dynamic context awareness is being realized in practice.

2.3.2.1 Context recognition overview

Context recognition is the process that senses, detects, obtains, and analyses context information from diverse sensory sources at runtime [107]. Context recognition is a subset of a broader context-aware and ubiquitous computing that seeks to identify user's contexts using data mining and machine learning techniques to learn patterns of user behaviors or situations. These learned patterns are used to induce predictive models to recognize users' future activities and contexts. In the literature, contexts can be acquired from diverse sources, generally called sensors. Sensors are categorized into two main types of sources: Software and hardware or physical sources [97]. In mobile environments, hardware or physical sensing involves using hardware sensors that are embedded as part of the internal hardware of the mobile device. Examples are accelerometer, gyroscope, GPS, compass, temperature sensor, etc. Software sensors are those that use applications to provide context information. Examples are calendar, social networks, browsing, and weather services. For context information to be of any practical use, it must be sensed, acquired, classified, or predicted in real time [99]. However, most existing recommendation systems only log context data, sometimes acquired explicitly from users, and process them offline to use in the recommendation process. This is not a realistic approach because it always requires user's interventions, which waste mobile user's time. Additionally, such approach

cannot be relied upon to provide dynamic preferences of mobile users at runtime. To acquire context information dynamically in real time, the traditional approaches rely on multiple sensors, wired to different parts of the user's body [105]-[106]. However, using wearable sensors to recognize user contexts is not realistic in practice because their outcomes have fostered little practical context sensitive applications [102]-[104]. Apart from being cumbersome to carry, these wired sensors are not flexible for building real life practical context-aware applications owing to their intrusiveness. On the other hand, mobile phones (such as smartphones) based sensor systems offer natural advantages over wearable systems because they are not intrusive and do not require additional equipment for data collection for accurate context recognition [97], [102].

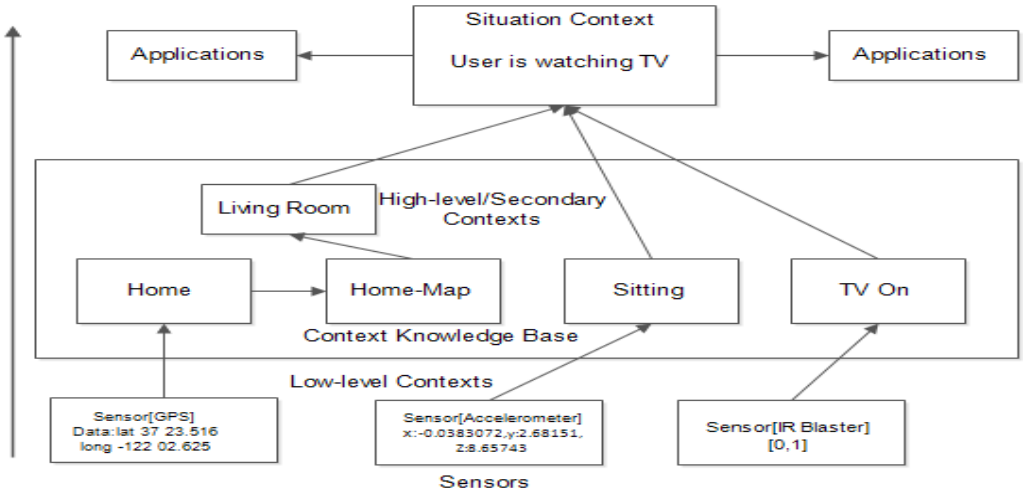


Figure 2.3- A three layer information flow to obtain high-level contexts from sensor's raw data

In addition, because they are always with the users anywhere, anytime, they can be used to obtain user's contextual information implicitly. However, using mobile phone built-in sensors has a major drawback: the mobile phone embedded sensors produce data that are in low-level forms, and are not suitable for mobile applications [38]. They are generally vague and imprecise and thus cannot be understood by the applications. Therefore, inferring meaningful context information from these data remains a challenge [110]. However, in the literature, to obtain high-level contextual information from low-level context data have been achieved adopting the processes illustrated in Figure 2.4. These processes are presented in next subsections.

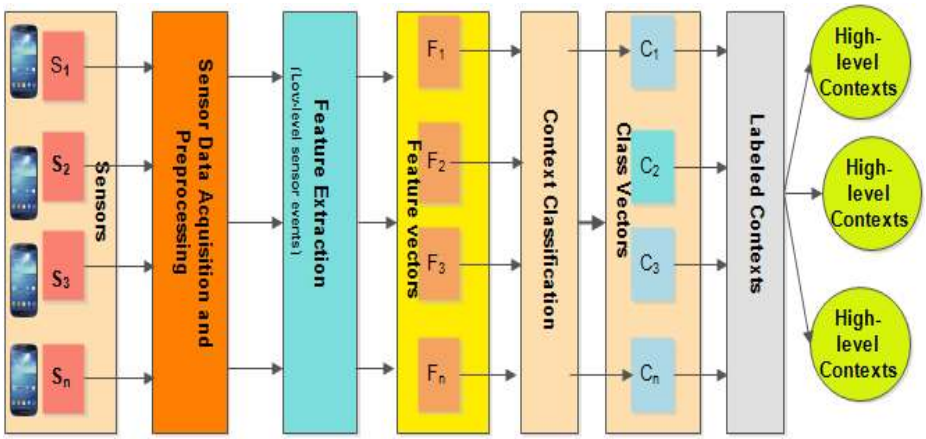


Figure 2.4- A generic architecture of context recognition processes

(a) Sensor data acquisition

Mobile phones with increasingly larger processing capabilities have become personal computing and communication devices for millions of users. Their ubiquitous nature and their capability to send, to receive and to process large amounts of data have opened up exciting opportunities in the area of ubiquitous service personalization. Moreover, these devices are now equipped with cheap built-in sensors, and are always close to the user even when not in active use. These sensors, range from GPS (for location context monitoring), image and video sensors (cameras), audio sensors (microphones), light sensors, temperature sensors, direction sensors (Magnetometer and Gyroscope sensors), and movement or speed sensors (Accelerometers, Rotation sensors) [112]. Smartphone built-in sensors now enable the development of powerful mobile applications [112]. Consequently, data provided by smartphone sensors can be explored to identify user's activities and contexts, and in recent years, this has become an important focus of researchers working in context and activity recognition [4], [38], [97], [105], [109]-[110].

Sensor data acquisition is an important step in the context awareness process. In this step, low-level data streams from sensors are captured for further processing in order to obtain meaningful high-level or secondary contexts. In most cases, physical signal values are the bases for measuring sensor event values. These values are usually sampled in time intervals, t, whose frequency has to be set to a desirable frequency of the sensors. Generally, a sensor vector $S_1 x S_2 x S_3 \ldots S_l$ defines sensor readings: $<s_1, s_2, s_3, s_l> t \in S_1 x S_2 x S_3 x, \ldots, x S_l$. Depending on the sensor type, the values $<s_1, s_2, s_3, \ldots, s_i>$ can be obtained in one or multiple dimensions. For example, accelerometers are usually in three dimensions, namely x, y, z-axes. The data obtained from each of these axes are then used in the next process.

(b) Feature extraction

Low-level context data are usually too large, sometimes containing many redundant data elements, which usually produce context misclassifications. Therefore, to minimize the effect of redundant data, the acquired data are transformed into a representative set of features or feature vectors in a process popularly known as feature extraction [107]. It is used commonly in pattern recognition to extract useful hidden information from raw data. It transforms the entire raw data into a reduced representation set of data features [107]. The main purpose of the feature extraction process is to simplify and to transform the raw data so that they can be interpreted better into high-level meaningful information. This is done, using domain specific methods to yield multiple data per sensor called feature F of samples $f \in F$ as a function of time or frequency, known as time domain or frequency domain features. In the literature, two main methods have been explored to extract features from data, namely statistical and structural feature [110]. Statistical methods, such as time and frequency domains, as used in many context recognition systems [99]-[107], use quantitative characteristics of the data to extract feature patterns, whereas structural feature extraction process uses interrelationships among time series data to extract useful patterns [108]. Statistical features such as mean, standard deviation (SD), Fast Fourier Transform(FFT), root mean square(RMS), etc. are usually extracted from numerical time series data, whereas for nominal data such as images, structural features are employed.
The output of the feature extraction process is a set of feature vectors $F_1 x F_2 x F_3 x \ldots x F_{n,,}$ which defines samples $<f_1, f_2, f_3, \ldots, fn> t \in F_1 x F_2 x F_3 x \ldots x F_n$ for points in time t in the multidimensional feature space. These feature sets are used as inputs in the next process, called context classification.

(c) Classification

In machine learning, patterns are discovered from a set of examples or observation denominated instances [110]. Therefore, the interpretation of low-level context data to realize the move from the feature space to high-level information is achieved in the classification process. The features generated in the feature extraction process are fed into machine learning algorithms to produce the predictive models to infer the context information. The task of the classification process is to find common patterns in the feature space, which are usually called classes or clusters. In existing work, a feature vector of the feature vector space is assigned to one or more classes with some degree of

membership, i.e. the probability that a given feature vector belongs to a given class. Other works have also explored clustering algorithms for context recognition, example can be found in [113].

(d) Labelling

Usually, to present the recognized context information in a human readable form to users, descriptive names are assigned to a class or a combination of classes from the last process. Labelling features into classes maps the class vectors for a set of names N, where N is a set of user defined context names, usually strings. For example, the user activity contexts, such as walking, driving, running, sitting, etc., are assigned.

One of these context labels $nt \in N$ describes the currently recognized user context in point time t where n is the recognized context class and N represents the set of class labels.

2.3.2.2 Recent context/activity recognition efforts

Recent advances in sensing technologies and mobile devices have fostered interesting investigations in the field of ubiquitous and context awareness, especially in context recognition [38], [97], [105], [109]-[110]. Many of the existing context recognition techniques commonly use multiple sensors wired to different parts of the user's body to execute context recognition task. In this section, the chapter analyses existing activity recognition approaches, focusing on the types of sensors used (wearable and smart device based), the processing techniques, the algorithms, strengths and weaknesses.

The traditional context recognition systems usually used sensors wired to the bodies of people. In [114], for example, Lin Bao and Intille used five 2-D accelerometers wired to the hip, upper arm, thigh, wrist, and ankle of 20 users to collect low-level data. They extracted mean, energy, entropy, and correlation features of these data, and then trained four classifiers namely, Decision Trees (DT), Decision Tables, Naïve Bayes (NB), and k-Nearest Neighbors (k-NN), with the DT achieving the highest activity context recognition rate of 80%.

In [115], Consolvo et al. present UbiFit Garden, which captures the levels of physical activities of the user and relates this information to their personal health, encouraging them to be physically active. Ubifit consists of three modules. First, a low-level module that continuously gathers relevant information about the user's activities, using a wearable 3-D accelerometer and a wearable barometer. Second, its feature processing and selection module processes raw sensor data into features for discriminating between activities. Third, it has a classification module consisting of a set of boosted decision tree classifiers, using the generated features to infer user's activities. Miluzzo et al. [116] present a multi-sensor system called ConceMe. ConceMe used user's contexts, such as walking, running, sitting, and standing to automatically retrieve and publish user's presence on social media networks. They extracted mean, standard deviation (STD), and the number of peaks per unit time (NPUT) features from raw data collected from accelerometer, microphone, camera, and GPS sensors of ten test users. They trained DT based classifiers to infer contexts from the extracted features. Their model can recognize most of the contexts. Nevertheless, it could not accurately distinguish between some contexts such as sitting and standing.

Davide et al. [107] present a survey of feature extraction techniques for context recognition. Using data collected from a single wearable accelerometer by one participant, they reported the performance of various feature extraction methods categorized into time, frequency, and discrete domain techniques. In their experimentation, three activities namely, jumping, running, and walking, were evaluated. However, this work provides a detailed performance evaluation of different feature extraction techniques. However, details of algorithms used for activity recognition were not reported. In addition, they trained and tested the system using data from a *Wiimote* sensor, wired to the body of only one participant, making their model user dependent.

Recently, attention has shifted to mobile phone's built-in sensor for context recognition. These mobile devices such as smartphones come with cheap, but powerful built-in sensors notably, GPS, cameras, microphones, and accelerometers, enabling to sense user's contexts [109]-[110]. These contextual data can be used implicitly to personalize the delivery of services to mobile users. Work such as those reported in [102]-[103] used smartphone accelerometer data to

recognize users' contexts. With this progress, the development of various applications for health monitoring, assisted living, content searching, social networking, environmental monitoring, and transportation has become reality [117].

A study presented by Mashita et al. [149] describes an interesting solution for recognizing user's activities. The system switches between two kinds of content search systems, according to the user's activities. The location-based content search system runs when the user is standing, whereas the menu-based content search system runs when the user is walking. Both systems present information that is adapted to users' identified activities. To achieve the context recognition task, they collected low-level data from the accelerometer, GPS and digital compass.
Extracting Fast Fourier Transform (FFT) features and using 128 samples, SVM algorithm was trained to classify user activities. They used cross-validation method to evaluate the classifier, but no details of the model's performance are provided.

In [97], Kwapisz et al. present a well-cited work that uses an Android based smartphone to collect low-level data from a single 3-D accelerometer. The low-level data were collected from 29 individuals while performing jogging, running, climbing stairs, walking, sitting, and standing activities. Six statistical features namely, average, absolute difference, average resultant acceleration, time between peaks, and binned distribution were used to generate further 43 summary features from 200 raw accelerometer readings. Based on these features, three classifiers namely: DTs, Multilayer Perceptron (MLP), and Logistic Regression were trained. With 10-fold cross-validation, the algorithms were evaluated. They argued that MLP achieved an overall best recognition rate of 98.3% for jogging activity. However, their models performed poorly to recognize climbing (Up and Down) activities.

Lau et al. [118] also investigated the suitability of Nokia N95 smartphone built-in accelerometer to recognize user's contexts by comparing the influence of window lengths on the classification algorithms' context recognition accuracies. Features such as mean, standard deviation, mean deviation, and standard deviation of FFT were evaluated based on 10-fold cross validation method, using DTs, Naïve Bayes, Bayesian Net, KNN, and SVM classifiers. Their conclusion is that average and standard deviation with KNN algorithm achieved the best recognition accuracy of 99.27%. Though this work is similar to CAMR's context recognition framework, it demonstrated the suitability of only accelerometer sensor for context recognition, whereas CAMR demonstrates the suitability of two additional smartphone sensors, namely orientation and rotation sensors.

Stefan et al. [104] report another interesting work that used data from an Android-based smartphone accelerometer and orientation sensors to recognize "simple" (walking, jogging, lying, running, etc.) and "complex" (washing hands, cooking, sweeping etc.) activities. Mean, maximum, minimum, standard deviation, correlation, and zero crossing features were extracted from different window lengths to train and to test six classifiers namely, MLP, Decision Table, Best First Tree, KStar, Naïve Bayes, and Bayes Net. The training and testing were executed using 10 fold cross-validation. For simple activities, they reported high recognition accuracy of over 90%, whereas complex activities achieved poor recognition accuracies. One interesting aspect of this work that is similar to CAMR is its investigation of the suitability of orientation sensor for accurate recognition of user activities. The study concluded that using accelerometer and orientation sensor improved recognition accuracies between 10-12 %.

Finally, on context recognition, the above cited research efforts explored either multiple or single dedicated wearable accelerometer for recognizing user activities, whereas others have relied only on mobile device accelerometer sensors. Even though the context recognition explored in this thesis is similar to the works discussed in this section, it differs in the number and type of sensors. For example, whilst most of the work relies on data from wearable accelerometer, specifically made for those research purposes, this thesis investigates the possibility of using accelerometer, rotation vector, and orientation sensor data in addition to the GPS sensor of commercial smartphones to recognize the mobile user's contexts accurately. The context recognition solution explored in the thesis is also different in that it evaluates the capability of various classifiers to identify some important activity contexts, using different window lengths and features. Additionally, the work differs in the area of intended application because its application is demonstrated in the area of context-aware personalized media content recommendations.

2.3.3 Context modelling and inference

In pervasive and ubiquitous computing environment, users always desire information that suit their contextual situations [1], [45], [73]. This necessitates the need to provide descriptions of contextual information, such as time, preferences, location, activities, noise level, illumination etc., in which users access information. Context awareness plays a primary role, helping such systems to know about the user's contexts and to act according to the user's interactions. Context awareness allows personalization of user environments and content consumption, thereby minimizing user's interaction with their devices and the provided services. However, the use of contextual information in these processes relies on the observation and representation of contexts. In other words, it depends on the context model used.

Similarly, this means that the utilization of contextual information in personalization and recommendation processes depends on the context model used. If the model is not properly designed, the derived context information may lead to inadequate or wrong information about the user's situation. Obviously, context representation model is an important precondition of any context aware system. This is an important process because the context consumers, i.e. the context-aware applications must understand and interpret the context information correctly in order to adapt accordingly. The correct observation of the context information retrieved from various sensors is the first step. The second step is the accurate interpretation of the information, which involves reasoning and inferences to derive contextual situation. In the last section, the chapter discussed the processes involved in the monitoring, acquisition, and recognition of contextual information from sensors' low-level raw data. In this section, the overview of the requirements and techniques to represent and to reason about the recognized context information from the perspective of media item recommendation in mobile environments is presented.

2.3.3.1 Context representation requirements

Traditional context modeling and inferencing approaches have been around for more than a decade [94]. These approaches have addressed, to a greater extent, the representation of context data within a model for consistency checking as well as to ensure that sound reasoning can be performed on the acquired context data [98]. In this regard, many context-aware applications have been developed based on various context models and in various application domains [119]. As analyzed by Bettini et al. [96], the development of various context aware applications has produced some basic requirements that contextual models must meet.

(a) Timeliness

This is one of the most important requirements of any context-aware application. Context-aware applications must be able to detect, capture, analyze, and infer context information and act accordingly in real time or near real time. For example, in a proactive context recommendation application that suggests delicious lunch of the day to a busy professional, it would be a bad recommendation, if such suggestion was delivered while he was in an important meeting in the morning. Existing contextual recommendation systems have always assumed this requirement because they expect the user to supply manually the contextual information.

(b) Heterogeneity

Context-aware recommendation systems have to cope with large variety of context information sources such as sensors, which differ mostly in their update rate and semantic level. For example, with mobile devices' embedded sensors, the combination of data from these different sensors must be acquired and processed in such a way that meaningful high-level context information can be obtained seamlessly. These systems must cope with dynamic and static context information streams. For instance, context history stored in a database, and the dynamic real time context information obtained at runtime must be managed to be able to infer present or predict future context information.

(c) Context dependencies

In practice, it is not uncommon to have one context, depending on another context(s) for the application to infer semantically meaningful high-level context information. For example, to infer user's activities such as "*walking*", "*driving*" or even location, this information relies on the raw data obtained from sensors. Additionally, to infer that a user is "*sitting at home*", it requires careful inference from more than one sensor data, which depends on the other sensors to arrive at that contextual conclusion.

(d) Context provisioning efficiency

In a distributed and mobile environment, efficient provisioning of context information is an important requirement. Accessing many of the existing context management systems have been a problem because they have been developed as legacy applications, which cannot be accessed from the web, for example. It is important also that each context information can be accessed via a well defined identification, based on the primary contexts such as the identity of the context objects, location, time, or user activities.

(e) Context reasoning

Perhaps the most important requirement is the ability of a context-aware recommendation system to derive new context facts from the acquired context facts or to obtain from the high-level context abstraction, a real world contextual situation. For instance, to conclude that *a user is seated in the living room watching TV* is a real world situational context that an intelligent context-aware application should be able to arrive at, given some primary and high-level contextual information. Additionally, a truly context-aware application must be able to reason about what to do when contextual changes occur. Besides, it must also be able to execute consistency verification. A context model must ensure that its formal or informal definitions correctly implement its requirements and functions in the real world [120].

2.3.3.2 Context representation models

Some important context models have emerged from existing context-aware management systems, based on the requirements presented in the last section. The existence of these well designed models has eased, to a large extent, the development of context-aware applications. However, they differ in their expressiveness power, in their support to provide reasoning on context information, and in the required computational reasoning performance [96]. In this section, popular context models generally found in the literature are introduced. They are roughly categorized into six categories, and their strengths and weaknesses are summarized in Table 2.2

Table 2.2- Comparison of context models

Model\Requirement	Expressiveness	Simplicity	Timeliness	Heterogeneity	Efficiency	Dependencies	Reasoning
Key-Value	x	√	x	x	x	x	x
Mark-up model	x	√	√	x	x	x	x
Object Oriented	x	x	√	√	x	√	x
Standard-based	x	x	√	√	√	x	x
Logic	√	x	√	x	√	√	√
Ontology	√	x	x	√	√	√	√

(a) Key-value pair

This model represents contexts in the most simplest form, by defining attributes of a context entity and its corresponding values. This type of model is very common in context-aware recommendation systems such as the models described in [15], [20], [73], [86].

(b) Mark-up model

This is a model using tags with attributes and content to represent contextual information in a hierarchical data structure. They are generally based on markup languages such as *XML* and *RDFs*. Composite Capability capability/ preference profiles (CC/PP) and User Agent Profile (UAProf) are good examples of this model [121] .

(c) Object oriented context models

This context modelling technique uses object oriented design paradigm to explore the power of object orientation such encapsulation and inheritance to represent context information. To access the context information, a well defined user interface is used. Yeung et al. [122] used object based model in a context-aware peer to peer news recommendation system.

(d) Logic based context Models

This technique uses facts, expressions and rules to define context information. It then uses a logic based system to manage context facts and expressions such as its addition, updating and removal.

(e) Standard based models

Various standard tools for context representation have also been developed in the last decade. Tools such as Composite Capabilities/Preference Profiles (CC/PP) [121], UAProf [123], CONNEG [84] and MPEG-21 [6]-[7] are the popular standards for representing contextual information. MPEG-21 Multimedia Framework (ISO/IEC 21000) is a more complete standard, which aims at defining a normative open framework for multimedia delivery and consumption for use by all players in the delivery and consumption chain [11]. Part 7 of MPEG-21, the Digital Item Adaptation (DIA) [14], provides several tools, among which the Usage Environment Description (UED) tool, for describing user information, natural environment context, and network and terminal characteristics. It also provides tools for adaptation decision taking. Some authors, to personalize and adapt media content, have used these standardized frameworks. An example is the work presented by Tseng, which uses MPEG-7 and MPEG-21 for personalizing videos [124]. Steiger et al in [125] also uses MPEG-21 for content delivery. However, these approaches lack the capability to establish relationships among low-level context. Effectively relating one context information with another could enable the provision for more effective and reliable recommendation decisions. These standardized context representation schemes, such as MPEG-21, do not provide the rich semantic description needed to support and to establish relationships between low-level context information [126].

(f) Ontology based context model

Ontology based context model uses a description of context concepts and relationships to represent context information. One major benefit of the ontology based context model is its natural support for reasoning. Additionally, the ontology based model can automatically derive additional knowledge from basic contexts. It can detect possible inconsistent context information to avoid erroneous context inferencing. Based on its simpler representation formalism, interoperability, heterogeneity, and the other benefits, ontology based context model is more favored than other context modelling techniques, and have been explored in various context-aware systems [18], [41], [96], [127]-[128]. In addition to their superior reasoning capability and expressive formalism for representing complex context data, ontology based context models have the natural capability for knowledge sharing because they provide a formal specification of the semantics of the context data [96]. However, despite the success of ontology based context model, it has been criticized for its lack of ability to derive new knowledge through facts. In addition, it is difficult to use OWL (Web Ontology Language) based ontologies, for example, to describe complex rules, to satisfy multiple conditions at the same time, to infer new knowledge [96], [129]. To address these challenges, Semantic Web Rule Language (SWRL) has been proposed. In [129], for example, SWRL and OWL were combined to provide an inference of high-level context information from multiple context data. Andrade et al. in [127] developed Multimedia Context-Aware Ontology (MULTICAO), which consists of a set of ontologies and SWRL based rules designed to provide

adaptable delivery of rich multimedia content in mobile environments. Yu et al. used ontology to infer situational context from low-level sensor data and user preferences to generate media recommendations for smartphone users [18]. They used ontology model to capture user's contexts such as location, activity and time to infer the user's preference.However, in CAMR, ontology model is used to infer high-level contexts from contextual information obtained from the user's smartphone. The inferred high-level context information cn then be used in the user profile model and the personalization processes.

In summary, existing context-aware recommendations systems do not generally meet some of the requirements as discussed in this section. For example, the contextual model presented by Adamovicius et al. [15] uses a hierarchical representational model capturing specific types of context attributes and their values. This model can be classified as a key-value context model and this type of contextual model lacks the capability: (1) to capture a variety of context types.(2) to capture dependencies and relationships between different levels of context abstractions.(3) Timeliness: they lack the ability to capture real-time context data and relate them with historical or static context (4) to check consistency of contextual information and (5) to enable context reasoning to obtain a real world contextual situation.

2.4 User profiling

A user profile is a description of an individual user that contains the most important or interesting facts about her. It contains the collection of all data about the user, especially his preferences, including general user data such as name, address, service specific data, context dependent information, user history, and other information that is inferred from other data sources about the user [56], [130]. Executing the collection of user data, its representation, and extracting useful information from the profile to personalize and recommend information item, such as multimedia content, or updating of the user data in the profile to keep up with the changing user information, is what this thesis refers to as user profiling. Because every user is unique and has different interests, tastes, preferences, background, and goals, to offer relevant and suitable media items to users, it is very crucial to learn the information about the user and to represent it in a form suitable for personalization or recommendation processes. This section analyses user profiling process, which serves as the core element and precondition of any personalized recommendation system, and reviews the traditional techniques explored in existing systems.

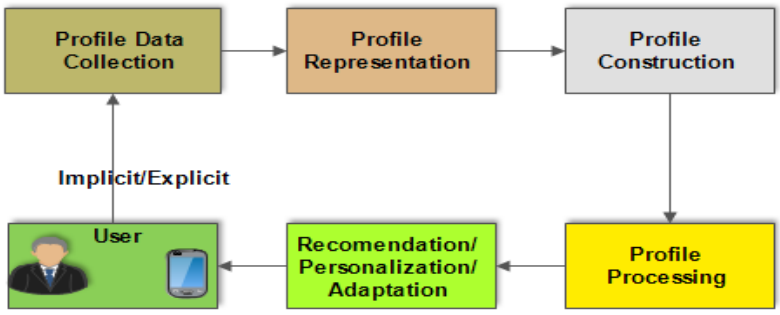


Figure 2.5- User profiling process

2.4.1 Traditional user profiling processes

For personalized systems, such as recommendation systems, to deliver effective and interesting service to users, it needs to learn information about users, such as their interests/preferences, optional demographic information such as name, gender, etc. User profiles can represent the preferences of either a group of users or an individual user. Generally, user profile can be classified into two broad categories, based on how the user profile information is sourced [10]. First, explicit user profiles are those that are directly solicited from users. Second, the implicit user profiles are

those that depend on software agents to collect user's information. Often, user profiles are constructed from information that is of interest to users, but information not preferred by users can also be used to build their profiles [56].

User profiles can also be classified on the basis of the *dynamicity* of its constituent data. User profiles that can be modified or augmented are considered as dynamic user profiles, whereas static user profiles maintain their data over a period of time without any changes [56]. Dynamic user profiles may be short-term or long-term user profiles. In contrast to the long-term user profiles, short–term user profiles represent the present user preferences that are not subject to change for a period of time. For instance, if a mobile user who enjoys watching movies with her family every weekend suddenly falls sick on a weekend and is hospitalized, her content preferences that weekend while in the hospital, would be different from her usual weekend preferences. In this example, her weekly movie events with family can be regarded as long-term profile information, whereas hospitalization related events can be regarded as short-term profile information.

Generally, a typical traditional user profile contains two dimensional information (2D) about the user: the user and his interests. This information is formally defined by Adomavicus et al. [15] as $R: User \times Items \rightarrow Rating$. Where R is the rating function, *User* and *Items* are the user and item domains. The rating expresses the degree of user's interests for the items. The rating is a totally ordered set of non-negative integers or real numbers within some certain range. To recommend items to users based on the user profile model, a recommendation system attempts to estimate the function *R* for the (user, item) pair. Once *R* is estimated for the entire User x Items corpus, the recommender system, then suggests the k-highest rated items for the user. The purpose of the user profiling process is, therefore, to collect information about a user's interest or preferences and the length of time over which they have exhibited such preferences in order to improve the quality of multimedia content delivery as well as user experience. Figure 2.5 shows a loop that represents the user profiling process. Next, each of the processes in the figure is described.

2.4.1.1 Profile data collection

This is the first step of the user profiling process where individual user is uniquely identified and information about her is collected. In literature, users have been uniquely identified through five basic methods namely, cookies, logins, proxy servers, software agents, and session ids [56]. Some of these methods, especially logins, software agents, and proxy servers provide accurate identification of users but are mostly invasive because they require the intervention of the user for identification. In most cases, users are always not willing to provide this information. In contrast, cookies and session ids are not invasive because they do not require the intervention of the user. However, they can lead to false identification or sometimes leading to profiles that are temporary. For example, if a user turns off his cookies or clears the cookies, the information about the user profile will be lost. For session *ids*, this provides a temporary user profile because each time the user uses the system, a new session id is created and if many users use the system, then it becomes impossible to track specific user for her profile information.

To collect user profile information, the user can be asked to explicitly provide the information or to implicitly implement agents that can collect the information on behalf of the user without her intervention. In most cases, a hybrid approach is implemented, where explicit and implicit user information is collected.

2.4.1.2 Traditional profile representation

Generally, a user profile is the representation of information about an individual user or a group of users that is used by personalization systems such as recommendation systems to provide adaptive and relevant information that meets the interests or preferences of that specific user or a group of users. It is basically a collection of information that describes a user [131]. For a recommendation system to understand the interests of an individual user, those interests must be represented in a form that is understandable to the RSs. Traditionally, there are three major user profile representation approaches.

(a) Weighted keyword representation

The weighted keyword profile representation is the oldest and the most popularly used technique for representing user's interests because of its construction's simplicity. It extracts implicitly key terms of the document content and associates weights with each term. It can also be explicitly solicited from the users. The weights associated with each key term are the numerical representations of the user's interests in the keywords. Each keyword or term can represent a topic of interest to the user or they can be grouped into a category to represent a more standardized interests of the user [56]. PRES [55] is a good example of personalized systems that uses this representation model. It uses content-based filtering techniques to suggest small articles about home improvements to users, using the weighted profile representation technique. Using the popular *tf-idf* [10], [46] weighting scheme, PRES associates weight w_i to each term t_i in document d. Another example is the Fab [66], which is a Web page recommendation system that explored the weighted keyword representation for user profiles. Other examples can be found in [56]. Though this method is very popular, its weakness is that it requires large quantities of user feedback to learn user preferences. However, this weakness has been addressed using methods such as Vector Space Model (VSM) [7]. Another problem is that a keyword or a term can have many meanings, making it difficult to accurately capture the semantic interest of the user. This problem can be addressed by the weighted semantic network user profile representation approach.

(b) Weighted semantic network representation

Weighted semantic network user profile representation attempts to address the polysemic problem of the weighted keyword representation by representing each keyword or terms as nodes representing concepts, and arcs are created based on the co-occurrence of two words/terms in those concepts (connected nodes) [56].

(c) Weighted concept representation

The weighted concept-based profile representation is similar to both weighted keyword and weighted semantic network representations. It is similar to the semantic network because both are represented based on conceptual nodes and the relationships between those nodes. However, unlike the weighted semantic network, the nodes represent abstract topics considered preferred by the user rather than specific terms or a set of related terms. On the other hand, it is similar to the weighted keyword representation because it is often represented as vectors of weighted features, however the features are concepts rather than terms or set of terms [56]. To express how much the user prefers a topic, some techniques are used. For example, numerical, boolean or weighted values are associated with each topic to express user's preference for each topic. There are many good examples of systems based on weighted concept representation. For example MyNews system [46] uses hierarchically structured weight based user profile model. It consists of two layers: One layer for fixed concept, which is common to all users, and a second layer, which contains dynamically generated events that are unique to each user profile. Other examples are ontology based user profile representations, which is more expressive, providing more semantic representation of a user's interests. Examples are those presented by Vallet et al. [132] and Sutterer et. al. [133]. The only weakness of this approach is its representation complexity.

2.4.2 Context-based user profiling

The traditional user profiling techniques discussed in the last section focus only on the interests and how much users express such interests, and not on the contexts in which users express their interests. These techniques cannot be used in mobile environment because user interests in mobile environments change, as users move from one place to another [130]. Therefore, to provide personalized and relevant information in mobile environments, it is very important to consider on the one hand, the contexts of use (which provides the description of the conditions, e.g. temporal, spatial, hardware, etc. in which the user accesses the information systems) and on the other hand, the preferences, which express what the user would like to obtain considering the contextual information [130]. Context-aware user profiling or contextual user profiling is a new user profiling paradigm that emerged from ubiquitous/pervasive/context-aware computing and personalization/recommendation systems to address the weaknesses of the traditional user profiling

techniques [130]. In literature pertaining to context-aware recommendations, Adomavicius et al.[15], [23],[73] were the first to formally propose the idea of context-aware user profiles, where they defined an extension of the traditional profiling techniques for recommendation systems. The traditional user profiling techniques explored two dimensional based techniques that consider only user and her interests. On the contrary, context-aware profiling techniques consider a multidimensional approach, accommodating user, her preferences, and contexts where user expresses such preferences.

Generally, a context-aware user profile contains three dimensional data. $R: User \times Items \times Context \rightarrow Rating$ where *User* and *Items* are the domains of the user and items respectively, and context specifies the contextual information that defines the user and the item domains. Based on Adomavicius' proposal, context-aware user profiling can be grouped into three main categories.

2.4.2.1 Contextual pre-filtering profiling

In the contextual pre-filtering profiling approach, the contextual information such as user location, time, mood, etc. are used to define the information contained in the user profile. The user and item in the traditional profile approach are now characterized by the context information. As shown in Figure 2.6, in the profiling process, the context information is used to define the 2D (user and items) information, which are relevant to the user. This approach is very flexible because it allows the deployment of the traditional profiling techniques after the pre-filtering has been executed, i.e. after a value has been assigned to the context variable and all data related to that context value about the user have been collected. The pre-filtering approach, according to Adomavicius et al. [8], uses the reduction process to obtain user's preferences in contexts. The output of the reduction process is still a 2D information, but it has been filtered according to the contextual information. As an example, take the 5-D contextual user profile of a user say, Ana, containing records of movies she watched in the past as well as weekday, time, and company (those with her when she watched the movies). Supposing, based on this information in her profile, she likes to watch romance movies on Friday, at 10pm when with her boyfriend, then the pre-filter assigns "*Friday"* to the *Weekday*, "10 pm" to the *Time*, *Boyfriend* to the *Company* contexts. In the pre-profiling process, only records (ratings, items etc.) with those assigned values are processed.

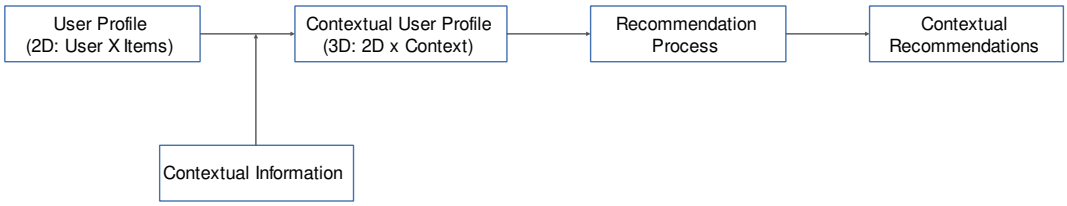


Figure 2.6 -Contextual pre-filtered profiling

The output can be fed into any of the traditional user personalization models without effecting any change. In fact, this process reduces the computational load of the traditional personalization process because all irrelevant records in Ana's profile would have been filtered and would not be considered in the recommendation process. For example, any record with Weekday = "Monday" becomes irrelevant! A good example of context-aware systems that uses pre-filtering profiling process is presented by Baltrunas and Amatriain [134], where the user profile is split into smaller profiles called micro-profile, through a process they call micro-profiling, on the basis of time contexts (e.g. morning, afternoon, evening, etc.). They applied this profiling method to personalize music recommendations based on the music a user prefers in time contexts. Other examples of context-aware recommendation systems based on this contextual user profiling technique are presented in [130], [135]-[136].

2.4.2.2 Contextual post-filtering profiling

The post-filtered profile, like the pre-filtered profile does not consider contextual information as part of the user profile information during filtering process. The basic idea of post-filtering process is that it keeps the 2D user information

in the user profile. During personalization or recommendation process, it uses this 2D information to personalize and generate recommendations. It then adjusts the generated recommendations based on the contextual information, as shown in Figure 2.7. However, the filtering still excludes the irrelevant records in the recommendation process, and prioritizes the recommendations based on the contextual information.

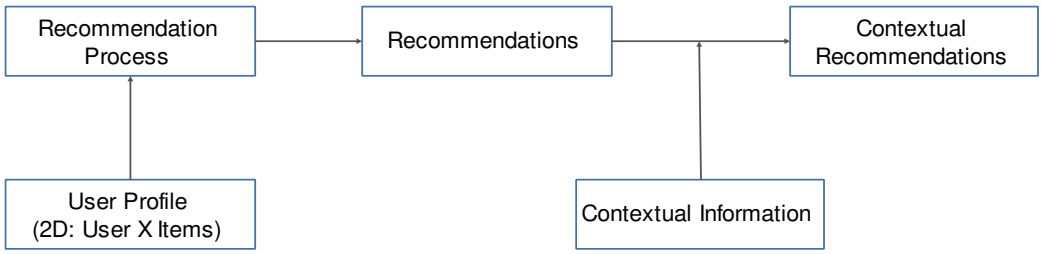


Figure 2.7 - Contextual post-filtering

There are very few systems using this profiling approach, but Cinemappy [137] is a very good example, which used post-filtering as well as pre-filtering profiling techniques. Cinemmapy uses pre-filtering based micro-profiling to create sub-profiles on the basis of time the user prefers to watch certain movies. After the recommendation process, it then uses location information to filter the recommended movies and presents the *k*-most relevant movies to the user. Like the pre-filtered profiles, the post- filtering has the advantage that it allows the continued use, and the deployment of the traditional profile models, i.e. easier integration of context information into the user profile and reommendation processes.

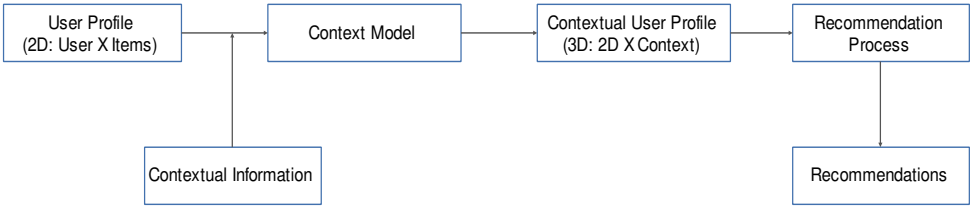


Figure 2.8- Model based contextual profiling

2.4.2.3 Multidimensional contextual profiling

Model based contextual user profiling, also known as multidimensional (MD) user profiling, is the only contextual profiling process that truly integrates contextual information into the user profile [8, 86, 163], and uses it as input in the recommendation process. Unlike the pre-filtering and the post-filtering processes, the model based user profiling process integrates $N \times D$ information in the user profile to personalize, and to generate recommendations for users. Each dimension D_i is defined by a set of attributes or properties A_{ij} ($j = 1, \ldots, k_i$), and these attributes are used as inputs to generate recommendations [see Figure 2.8]. For example, in content based recommendations, user's attributes (such as demographic information like age, sex, etc.) and item's attributes (genre, price, etc.) (2D) have been used in the traditional profiling process [10]. Adomavicius et al. [15] extended this profiling approach into a truly multidimensional profiling. A multidimensional profile is defined by three new sets of important attributes.

2.4.2.3.1 Derived attributes

Derived attributes include, for example, the average number of times a user has consumed an item of a certain genre in a certain place or time or the most visited location by a user, etc. These are more complex attributes derived from simple attributes of a user, his contexts, and consumed items.

2.4.2.3.2 Set of rules

The set of rules relates the attributes of the user, his contexts, and attributes of the preferred items, to derive a more meaningful characterization of the user. For example, we may describe the music listening interest of a user, say, Ana, as follows: "Ana prefers listening to pop music whenever she is on transit on weekdays". This rule relates the following attributes in Ana's profile: Name = "*Ana*", MusicGenre = " *Pop*", DayOfWeek = "*WeekDays*", Activity = "*In Transit*".

2.4.2.3.3 Signatures

The signatures define the data structures used to capture the evolving behavior of a user, which is earned from a large stream of data obtained from his transactions [15].

Finally, to realize the context integration process, the model based user profile process essentially explores predictive algorithms such as support vector machine (SVM) [138], Bayesian [79]-[80], Neural Network [81], case based reasoning (CBR), or multi criteria decision algorithms, etc. to include multidimensional context information, besides user and items, in the profiling process [50].

2.4.2.4 Summary of context-aware user profiling techniques

Multidimensional user profiles provide richer and better recommendations than other context-aware user profiling approaches because it allows much more contextual information as input in the recommendation decision making process. With MD based user profiles, a context-aware recommendation system can provide, for example, "10-top romance movies to Ana when in transit in a bus". It also allows targeted recommendations. For example, when at the airport, interesting restaurants in a destination country can be recommended to a travelling user.

2.5 Content adaptation in mobile environments

One of the objectives of Universal Multimedia Access (UMA) is to enable user's transparent access to any kind of content from anywhere, anytime, and with any device [6], [139]. Therefore, it is important that systems that provide access to multimedia content, such as multimedia recommendation systems, not only provide suggestion of relevant media items to users, but also should understand the provisioning contexts in order to present the recommended multimedia content with the best possible quality in specific contextual conditions. For the successful delivery of rich and adaptive recommendation of multimedia content, tailoring the presentation of the content, based on the limitations of the user's device and network constraints is an important consideration. This process has gained considerable acceptance in today's multimedia communication research communities as described in [140]-[145]. However, the advances in technology will continue to emphasize the heterogeneity in networks, devices, systems, software, and services [see Figure 2.9]. This will certainly increase the demand by multimedia services consumers for more choices, and the demand for better quality and more personalization options, which ultimately will complicate the design of such systems for dynamic and mobile environments [146].

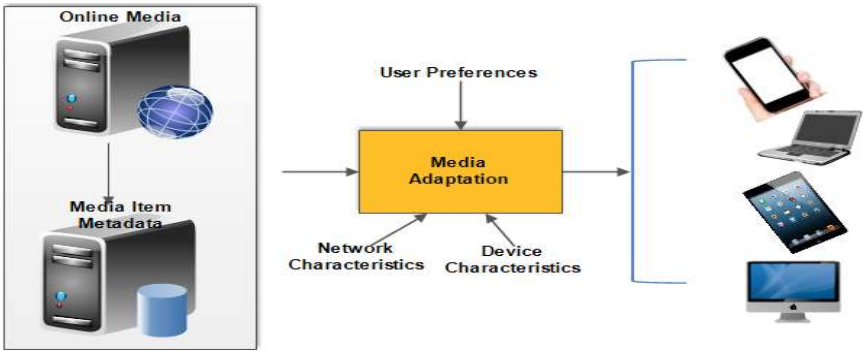


Figure 2.9- Typical media adaptation architecture

To satisfy multimedia content consumers with rich quality multimedia content presentation, therefore, the characterization of recommended multimedia content and the context of their presentation are needed to enable adaptable quality and hitch-free delivery of multimedia services.

Thus, constraint presented by the contexts of usage, the characteristics and conditions of usage, such as, network contexts (Bandwidth availability, error and loss rates, jitter, etc.); terminal contexts (screen size, CPU speed, installed software, available memory, etc.); surrounding environment (noise level, presence of other people, lightning condition, etc.); user (user preferences such as language, text over voice, images over video, etc., disabilities) must be taken into consideration [6],[127],[147]-[148].

In the multimedia content adaptation process, Adaptation Decision Engine (ADE) is responsible for taking decision on whether the content requires to be tailored to network condition and device characteristics [146]. The ADE collects contextual data and processes the data into knowledge for taking adaptation decisions. A significant number of parameters, each of which is likely to experience variations throughout the service lifetime, characterize the context of usage in networked multimedia applications. Acquiring and correlating measurements of these parameters is not a trivial task and are usually performed continuously or on a periodic basis. Therefore, the ADE as the intelligent part of the adaptation service must be able to acquire contextual data, interpret them and then select the multimedia parameters that best suit the prevailing contextual conditions, including user preferences, for increased user experience. Generally, as stated in [146], the adaptation decision process is seen as a problem of resource allocation in a generalized rate distortion or the resource distortion framework. The ADE, when fed with a set of available resources that are seen as constraints (such as available network bandwidth or terminal display size), should be able to select the set of service parameters that leads to a variation of the content that satisfies those constraints whilst maximizing a given utility. This utility is usually the content or service quality, but it could be other parameters, such as the service price. The association between the available resources (characterized through the acquired contextual metadata) and a set of multimedia service determines the resulting utility. Several approaches have been explored in the research communities to proffer solutions to the problem of multimedia content's poor quality due to the identified contextual constraints [127], [149].

Machine learning techniques such as Reinforcement Learning, Bayesian Networks, Neural Networks, etc. plays important role in designing self-adaptive and predictive capabilities [150]. These approaches have been widely used in Robotics and multi-agent systems as surveyed in [150], [151]. Recently, they have been considered to engineer adaptive multimedia systems [140]-[141]. They tend to explore fully decentralized and uncoupled coordination. They rely primarily on environment mediated local context information that translates into desirable properties such as scalability, and adaptability to changing conditions, dynamic scenarios and robustness to partial failure or disruption of normal activity. These algorithms have been incorporated into standardized multimedia adaptation frameworks such as MGEG-21[6] and MPEG-7[152]. For example, Andrade et.al [127] developed an advanced and robust multimedia content adaptation solution to provide interoperable delivery of adaptive multimedia content in different application scenarios. There are several other multimedia content adaptation built on top of MPEG-21 and MPEG-7 framework which can be found in [124], [128], [142]-[143], [146], [153]. Anastasios Doulamis and Geargios Tziritas in [141] presented a multimedia content adaptation decision technique based on neural networks for efficient video delivery over low and time varying networks. The key frame prediction was performed using the neural network model. This is an interesting work, although it is limited in contextual scope and the prediction algorithm only targeted the video contents. Another example is the system presented by Tayeb Lemlouma and Nabil Layaida in [154]. They developed a negotiation model for the decision process on the server side in order to match different context constraints to find an agreement between the server adaptation capabilities and the client preferences and constraints. D. Jannach et al [142] presented a framework for adapting multimedia systems underlying resources to device capabilities using a knowledge based decision engine. In [155], Pedro M. Ruiz et al used a genetic algorithm to optimize the quality of service of real-time multimedia adaptive applications and to determine when to trigger the adaptation decision depending on the network conditions. A set of encoding parameters that offers best user's QoS while satisfying the network condition was used but it only considered network contexts.

The systems described above adapt content that is explicitly requested by the users. They have no mechanism to implicitly request on behalf of the users, relevant multimedia content that is suitable to their preferences without user's interventions. In this thesis, the adaptation process is adopted as an optional and integral mechanism, to assist tailoring recommended multimedia content presentations to suit the device and network constraints.

2.6 Context-aware mobile media recommendation systems

From the late 1990's when recommender systems emerged, it has rapidly evolved into more than just a system for suggesting interesting items to users in traditional computing environments. From the traditional recommender systems to mobile and pervasive computing domains, context-aware mobile recommendation systems have emerged and have been proposed to solve the multimedia information overload problems, and to enable ubiquitous access to relevant multimedia contents on the Web using mobile devices [4], [30], [50]. Context-aware media recommendation systems can become the de facto solution allowing anytime, anywhere access to multimedia items in mobile environments by predicting and presenting *contextually* relevant multimedia information to mobile users. "*Contextual relevance"* adds a new dimension to the recommendation system problems, which involves the consideration of contextual information about mobile user during the recommendation process. In this section, the chapter provides the overview of some existing context-aware mobile recommendation systems, using contextual information as a precondition for providing media recommendations.

Copernik: Coutand presents Copernik, an interesting framework for developing personalizing context-aware personalized applications [156]. It includes a contextual personalization cycle that encompasses required set of operations involved in the personalization of applications. It uses case-based reasoning to infer user preferences in the present context from their previous contextual information to personalize applications, even in unforeseen contexts and with limited user interventions. Though our proposed work is similar to Copernik, however, one important difference between our approaches is that Copernik does not provide specific means to capture and process user's dynamic contextual information. Our proposed system provides real time context capturing, semantic representation and inference of dynamic user contexts using their mobile devices. Additionally, Copernik only personalized applications for users using context information, whereas our system provides personalized media items to users by incorporating user's dynamic contextual information in the personalized recommendation process. Copernik suffers from new user problem. For example, it only provides personalized applications for only those users who have used the system in the past. For new user, there is no provision. In our case, this problem is addressed by incorporating mechanism that uses the contexts of other users whose past contextual information is similar to the new user's present context information.

mPersona [43] is one of the earliest context-aware mobile recommendation systems, and was developed to provide flexible personalization for mobile users, taking into account their mobility, local environment, and user's device profile. It utilizes various characteristics of mobile agents to support flexible, scalable, and modular personalization that supports user mobility. mPersona delivers innovative solution to cellular subscribers, providing personalized content according to their preferences and environment information, taking into account not only their personal profiles, but also their mobile device profiles. With minimal clicks, mPersona provides users with the exact information they need thus, reducing access time and the cost of browsing using their mobile devices. For each user, it builds a wireless web portal according to the user's profile. It implements a flexible middleware and protocols to handle the diversity of content structures and their semantics. With a set of algorithms, it matches up the user profile with the local content, and dynamically personalizes the user's portal, adapting it to the user's device characteristics. To realize its objectives, mPersona implements three broad component types. The entry point components, which are responsible for user's registration and profile creation and managements. Second, the content side components, which manage content description and metadata structures. They also manage the content selection process and the user's personalized portal metadata. The third, the client side components reform, transform and deliver user's desired content. Although mPersona provides between 30% to 60% improvements in the recommendation accuracies over the traditional recommendation processes, however, unlike the research presented in this thesis, its major weakness is that

it only considers the device context in the user profiling process. The user's activities and other contextual situations were not taken into account when providing recommendations.

MediaScout is a commercial recommender system for recommending media content, such as movie trailers and clips, to mobile device users [57]. It uses a combination of techniques, such as expert systems, collaborative filtering, and content-based recommendations, in a hybrid algorithm, exploiting the advantages of various approaches while minimizing their disadvantages. MediaScout registers a new user through a manual questionnaire, which merges the new user to a group of existing similar users (called community) and then generates recommendations to the users based on the interests of the user community he belongs. Additionally, MediaScout was designed to deliver media content recommendations to mobile phones via a stereotype approach, which combines elements of both content based and collaborative filtering techniques to provide recommendations. The stereotype is a generalization of users, an abstract user entity that provides a general description of a set of similar users that form a community. The stereotype is described by a set of ratings over items, and user similarity can be identified by his affinity to various stereotypes. Recommendations are generated based on the stereotype after the normalization of the user's affinity to a stereotype. MediaScout also defines ontology model of the media items, defined by experts in the field. A media item profile is an instantiation of the ontology. The stereotype assigns relevance values to the various attribute values of the ontology. Generating recommendation based on the stereotype is done by matching item profiles with the stereotypes, resulting in relevance values over media items. Its user profile is modeled by a list of affinities it has with stereotypes. To initialize the systems, it uses a decision tree based questionnaire, which forces the user to choose one branch from seemingly related alternatives. Though this work is similar to this thesis, in that it was designed to provide recommendations for mobile phone users, however, MediaScout does not take into account, the user's dynamic contexts as well as the profiles of the device, network, and environment information.

The aceMedia by Vallet et al. in [1] is concerned with exploiting semantic and ontology-based contextual information, specifically aimed towards personalization for content access and retrieval. The system endows personalized multimedia management systems with the capability to filter and focus their knowledge about user preferences on the semantic context of ongoing user activities to achieve coherence with the thematic scope of user actions at runtime. It uses an ontology based representation model of context of ongoing retrieval tasks, which is used to activate different subsets of user interests at runtime, in such a way that out-of-context preferences are discarded. Using an ontology-driven representation of the domain of discourse, it provides enriched semantic descriptions of the retrieval process and preferences, and enabling the definition of effective means to relate preferences and contexts. It extracts and includes real-time contextual information as a means for enhancing the effectiveness and reliability of long-term personalization. In addition, it enables a more realistic approximation to the highly dynamic and contextual nature of user preferences, in a novel approach with respect to prior work. The work gained accuracy and expressiveness via the ontology-based approach ultimately bringing additional improvements in terms of retrieval performance. The limitation of aceMedia, however, is that it was designed as a retrieval system, which can only provide media items explicitly requested by the users in their contexts. Unlike CAMR, aceMedia was not designed to address issues of implicit provisioning of media item recommendations.

Another context-aware recommendation system is AMAYA [157]. AMAYA is a multipurpose recommender system, which provides a generic framework that delivers contextual recommendations based on the combination of previously gathered user feedbacks (e.g. Ratings, clickstreams history, etc.), contextual data and ontology based content categorization scheme. AMAYA was designed and built to meet two basic requirements. First, to provide recommendations for all situations that might arise in the user's life, by mapping all personalization data to specific situations. Second, to provide support for any number of services a user might want to consume by providing a generic interface to a number of learning algorithms and prediction methods as well as introducing ontology based content categorization scheme. AMAYA implements four basic components, namely: Data adapter, which enables the retrieval and processing of distributed personalization data by providing an interface that abstracts from the actual technology used to store personalization data. The profile manager groups the personalization data in terms of profiles,

providing mapping to one or more specific situations. The profile broker allows querying of all personalization data needed in specific situations. The recommendation supports learning of external preferences by providing a generic interface to multiple learning algorithms and prediction methods. AMAYA is well related to the work presented in this thesis because it also uses contextual user profiling approach to provide mappings between a user's preference and contextual situation to recommend relevant content information. It separates the contextualization from the actual recommendation process, delegating it to the profiling process. A good benefit of this approach is that, the traditional recommendation algorithm can be used without any modification. However, AMAYA only allows explicit mapping of user's preferences to contextual information. For example, the system cannot distinguish between the preferences of a user in the office or home situation except users explicitly defined their contextual preferences.

CoMeR [18] is a generic and flexible context-aware media recommendation framework based on the *NxM* multidimensional user profile model developed to recommend media items for smartphone users based on their preferences, situation contexts, and capability contexts. It considers contextual information ranging from user preferences and situation contexts to device and network capabilities as inputs for both content and presentation recommendations. It generates as output, a multidimensional recommendation, offering a content rating. It implements a hybrid recommendation approach that exploits content-based approach. The content-based approach evaluates the media items against user preferences. The Bayesian classifier utilizes situation contexts and computes the probability of a media item's suitability for specific environments, such as user's home/office, day of the week, and time of the day. The rule based method, using a set of if-then-else rules, filters the media items based on the capability contexts, which have been modeled and instantiated into an ontology-based context inference model. Only items, which satisfy user device and network capabilities such as supported formats, frame rates, modality, etc., are recommended. This system is similar to CAMR in that it also incorporates hybrid recommendation and contextual information to provide multimedia items to smartphone users. However, unlike CAMR, it is not clear how it obtains context information from the user's devices. Additionally, it relies on user ratings in specific context the recommendation process.

Reperio [74] is a flexible and generic industrial recommendation system, developed to deal with several kinds of data sources: user logs, catalogs of items with their descriptions, user preferences, item's attributes and social network data. It was developed using the kNN neighborhood method, which incorporates item-item based recommendation to suggest items to users. Reperio incorporates content-based, collaborative, and hybrid recommendation techniques into a multi-platform framework in a client-server based architecture. The server (Reperio-C) is a centralized system implemented as web services. The core modules of Reperio-C are the similarity matrix, which associates each item with a set of most similar items. With a scoring function, which is responsible for ranking the items according to their predicted ratings, it produces a set of items ranked in decreasing order of the predicted scores. The client component of Reperio (Reperio-E) is a suite of mobile application development libraries for implementing Reperio client on any terminal with Java Virtual Machine (JVM), which includes PCs and mobile Java based frameworks such as Google Android smartphones. Reperio is much related to the work presented in this thesis because, like Reperio, CAMR is built using neighborhood based methods, incorporating three core context-aware recommendation techniques. Additionally, like Reperio, the core recommendation functions are implemented as web services on a centralized server with mobile client implemented on smart device. However, unlike CAMR, Reperio is not a full context-aware mobile recommendation because contextual information such as user's activities, environment context information etc., which are related to improving recommendation accuracy were not taken into account.

Pessemier et al. present another closely related and interesting context-aware mobile recommendation framework in [26]. This is a generic framework with the capability to detect user's contexts and physical activities by analyzing data collected from the mobile device's embedded sensors. The framework was developed to provide personalized content of various categories such as point of interest (POI), train schedules, and tourist information based on the user's contexts and physical activities. To enable the use of community knowledge (such as data regarding user's behavior, feedbacks, and contextual information of all users of the system) in the recommendation process, personalized recommendations are computed based on collaborative kNN based technique on a central server. The client makes requests to the server by including the user's current contextual information and activity, returned from a context

recognition framework. Using the user profile model, the framework computes the most appropriate context-aware recommendations tailored to the user's current contextual needs. The user profile model is a rule-based model, which uses if-then-else rules to relate user preferences to their contextual information. The rule, which is manually created for a first-time user, is stored as a 4-tuples (user, context, category, score), where score indicates how important the rule is for the user. Similarly, it consists of various models such as history and popularity models, which generally are used to track the consumption of the user's content categories in different contexts. Though similar to CAMR, it differs from CAMR in terms of the types of sensors used to gather context information, the type of recommendation techniques used, and the update process implemented. In terms of sensors, whereas the cited work relies on only accelerometer to obtain user's physical activities, which are user orientation dependent. CAMR on the other hand, provides alternative sensors such as rotation and orientation sensors. In terms of recommendation techniques, since the proposed solution is designed to recommend different categories of content like the work presented by the authors, we extended and incorporated content-based, collaborative, and hybrid recommendations with contextual information. Our proposed solution makes provision for manual creation of the user's initial profile. However, it can handle a new user by using the new user's present context information to find others that have been in such context and the media items they consumed in those contexts to make initial recommendations, thereby addressing the new user problems.

MAGITTI is another interesting context-aware mobile recommendation system developed to recommend content that is related to leisure activities of a mobile user [158]. It automatically generates recommendations without user's entering a query for content matching their inferred activities. The activities are inferred from the sensed user contexts. Five basic activities are inferred by MAGITTI, which are *Eating, Shopping, Doing, Seeing, and Reading*. Its main goal is to infer these activities from the detected user's current contexts and then infers likely present or future leisure activities, which are then used to filter and recommend content such as movies, restaurants, parks etc. to the users. MAGITTI's main feature, which distinguishes it from other context-aware mobile recommendation is its ability to provide new items, which are not related to the items previously recommended to the user, a property of recommendation generally referred to as serendipity. MAGITTI implements a hybrid recommendation process, incorporating some important models. Its collaborative model calculates the similarities between users, using ratings and scores of each item based on how other similar users have rated it. Its preference model scores each item according to how they match the user's preferences (similar to explicit feedback model), whereas its learned preference model scores items based on the user's observed behaviors or feedback (implicit preference model). The content-based model measures the similarity of an item to the profile of users' previously viewed items. The distance model gives maximum weight to items, which are within the range of the user's location, whereas it uses an exponential decay function to rate other items. It uses a model called boredom buster to reduce the score for items that users have previously seen, thereby providing diversity and serendipity in the set of recommended items. Like MAGITTI, CAMR uses contexts and activities of users to recommend relevant and interesting media items. However, whereas, MAGITTI uses context data obtained from GPS and device calendars to infer five activities, CAMR uses available sensors on the user's mobile device to infer physical activities combined with location, which are encapsulated in a dynamic contextual user profile model to learn user's consumption preferences in specific contexts. Like MAGITTI, the proposed system implements context-aware hybrid recommendation technique, but also incorporates other techniques such as context-aware content and collaborative recommendations. The recommendation techniques are designed as web services, and can be consumed on any mobile hardware or software platforms.

2.7 Summary

Recommendation systems have emerged as a solution to the information overload problem. However, traditional recommendation systems cannot be applied effectively in mobile environments. Existing recommendation systems in mobile environments explore limited context information to tailor the recommendations to users whose preferences change as they move from one place to another. Additionally, these systems explicitly provide recommendations to users, by waiting until a user makes a request before initiating the recommendation process. Other systems, especially in the multimedia adaptation domain, that adapt media item content that is explicitly requested by users to match the device and network constraints, are yet to fulfill the goal of the universal multimedia access (UMA) via an anywhere,

anytime mobile media consumption. The requirement of implicit delivery of media items that are relevant and, matching user contextual preferences and device and network constraints are yet to be realized fully.

This thesis proposes context-aware media recommendation framework for smart devices (CAMR). CAMR addresses context as a high-level concept that enables the consideration of a much more comprehensive description of real-world situations via a context awareness framework that integrates a combination of classification algorithm and ontological reasoning. Rather than using individual contextual values directly acquired from sensors, or explicitly requesting users to supply own context information, these data are dynamically acquired, from the user's mobile devices, processed and utilized to infer probable situations of consumption of multimedia resources. It acquires and captures low-level contextual data generated by mobile device embedded sensors in a predictive model based on a classification algorithm to produce as output higher-level context information, proposing the representation of contexts first with optional MPEG-21 UEDs, and then instantiate it into ontology and rule based model to infer contextual situational knowledge.

Finally, unlike the reviewed systems, this work proposes an original and quite flexible user profile structure, introducing the concept of contextualized user profiles, incorporating the context dimension whilst enabling easy update and usage. The contextual user profile model incorporates a hybrid of context pre-filtering and a context model approach, as user profile model and a kNN approach to relate user's present and past contextual preferences. CAMR also integrates contextual content-based, collaborative, and hybrid recommendation techniques using vector space models (VSM), with an optional media content presentation adaptation. Finally, this chapter has laid out the background of the research presented in this thesis in four main areas. Context-aware computing, user profiling, personalized recommendation systems and multimedia adaptation. The chapter presents how the research conducted in this thesis relates to and differs from other researches in context-aware mobile recommendation. In subsequent chapters, the solutions provided by the proposed framework will be presented in detail.

Chapter 3

Conceptual Framework for Context Awareness

3.1 Introduction

Context awareness is one of the primary characteristics of emerging mobile and pervasive applications. The ability of these applications to acquire user's contextual information, and to understand the meaning of such diverse data to offer users richer experiences is the realm of any efficient context-aware system. As discussed in chapter two, valuable work has been conducted in the last years to develop context-provisioning frameworks, some of which have been applied within the field of multimedia recommendation, addressing some of the key challenges in context awareness. Nevertheless, these solutions lack the feature that unifies context sensing, context recognition, context representation, and context inference, to foster their applicability not only to content recommendations but also to other domains in a dynamic and interoperable way. The next generation of context-aware applications are expected to interact with different types of sensors in real time, analyzing and validating the sensory information from diverse sources, associating this information with implicitly perceived user contexts and preferences, and reasoning about the contextual information to obtain rich and meaningful descriptions of contextual situations of the user. These requirements together with the dynamic nature of mobile environments, lead to high levels of complexity when designing such advanced mobile context-aware systems, which is a challenge that is yet to be addressed fully.

Therefore, a holistic approach is required to derive various types of high-level contexts, from different sensing sources, to support proactive and interoperable provisioning of high-level context information that can be shared and consumed by mobile applications. This approach requires that proper semantic information and concepts from the mobile multimedia domain be identified to enable the definition of relevant and most common contexts occurring in those environments. Additionally, it also requires the identification of appropriate low-level characteristics that maybe collected and subsequently processed to allow inference of such high-level contexts or situations dynamically. This technique needs to define user contexts by providing a mechanism to infer them, based on contextual situations of the users.

To this end, this chapter first presents a unified conceptual context-provisioning framework, which provides sensing, recognition, and inference of contextual information with support for the development of contextual applications, notably context-aware recommendations [8], [38]. Second, it introduces a five-component framework, its key components and its services. Such conceptual framework incorporates reasoning techniques that allow derivation of high-level context information from low-level sensor data. Third, it allows flexible number of recommendation services that can be supported by the context-aware personalized recommendation service. More recommendation services supported by the context infrastructure can be added or removed without affecting the overall functionality of the system.

3.2 Motivation

A large number of current systems were developed based on explicitly acquired contextual information. On the other hand, those using implicit context sources focus on using low-level atomic information obtained from wearable sensors. In fact, the majority of work conducted in this area focus on explicitly acquired contextual information and wearable sensors [112], [116], [159]; their outcomes have fostered little practical transparent context sensitive applications.

Apart from being cumbersome to carry, wearable sensors are more difficult to program, thus making it complex to build practical context-aware applications. On the other hand, smartphones' sensor systems offer natural advantages over their wearable counterparts: they are not intrusive and are reasonably easy to program with the availability of powerful and well-documented APIs [103]. Typically, they remain close to the user even when not in active use. This allows smartphones to be transparently used to monitor user contexts anywhere, any time. For example, such ubiquitous sensing ability of user contexts can be used to mute the device's ringing tone automatically when the user is in a meeting or to select between audiovisual and text presentation or to deliver relevant multimedia content to users proactively depending on the sensed context. Still, smartphone built-in sensors, just like any other physical sensor, have a major limitation: they produce data in low-level form, which are mostly not suitable for building advanced mobile context-aware applications. Thus, inferring meaningful contextual information from these data has remained a challenge.

Taking into consideration the study and analysis described in chapter 2, highlighting existing shortcomings and limitations, together with the hypothesis formulated at the beginning of this thesis, the main objectives of the conceptual context awareness framework developed in this thesis are to:

(1) dynamically recognize contextual information from mobile device's in-built sensors, encapsulating the contextual information into standardized descriptions and representations, notably MPEG-21 UEDs [160] and MPEG-7 multimedia description schemes (MDS) [152];

(2) generate additional knowledge from the sensed contextual information using semantic technologies, notably ontology and semantic web rules (SWRL) [129]. Such additional higher-level knowledge should be able to describe in a richer way the situation the user is in, combining atomic low-level contextual data, thus empowering the context-aware application to take better-informed decisions or actions;

(3) offer the set of context-aware functionality accessible via a Web Service interface, thus facilitating its use.

Therefore, the key distinctive feature of the proposed contextual framework is its ability to proactively and automatically provide high-level context information as a service, to be consumed by diverse external applications. Such high-level context information is inferred using classification and ontological models, which can also be accessed as services by other applications. Figure 3.1 shows the five main components of the conceptual context-provisioning framework.

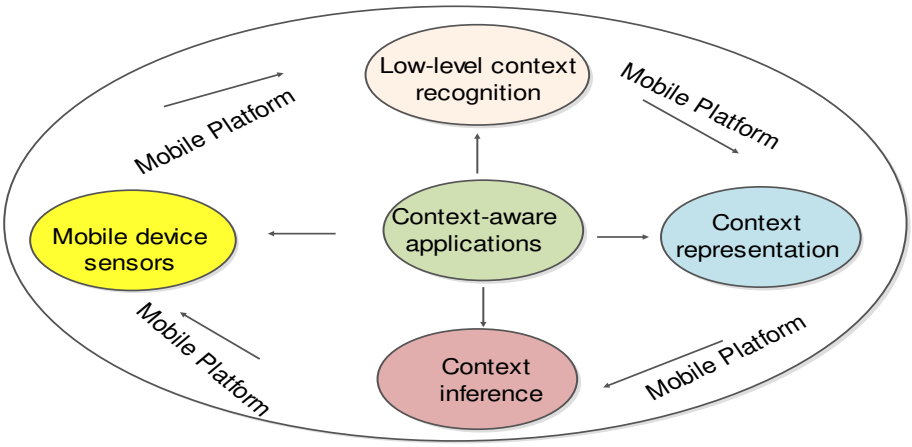


Figure 3.1- High-level architecture of the context awareness framework

The first two components (mobile device sensors and low-level context recognition) focus on acquiring contextual data and preparing it for further processing. They are responsible for sensing and monitoring contextual information using the device's built-in sensors, and for pre-processing and filtering it, obtaining adequate features that will be used

subsequently to recognize meaningful contextual information from the sensed low-level context data. The third component focuses on context representation. This component allows sharing and/or integration of contexts between different context sources, by using standardized descriptions for representing and packaging the acquired contextual information. For this purpose, MPEG-21 UED, and MPEG-7 MDS standards have been adopted, offering a standardized context representation model, which together with a Web Services interface, enables its universal use.

However, because such standardized schemes lack: 1) the capability to reason on the context information to infer additional high-level information; 2) the capability to describe context concepts and their relationships and; 3) the capability to check the consistency of the contextual information, the proposed framework includes a fourth component focusing on context inference, utilizing semantic and classification technologies. It includes a machine learning classification model and an ontology contextual model, which provide contextual recognition ability, reasoning and inference. The fifth component of the framework is the application layer that exposes an interface for context-aware applications to access the contextual services provided by the framework, implemented by the other components. All parts of the framework rely on a central knowledge base, where data and information about user contextual information are stored. This layer provides an interface for applications consuming the provided contextual information. Unlike existing context-aware frameworks for mobile recommendations, which are essentially reactive, i.e. they act only when an explicit recommendation request is issued, and rely on explicitly provided contextual information [10]-[11], [19], we proposed a context-aware framework to dynamically and transparently provide high-level context data to the recommendation processes. In explicit context frameworks, users are expected to provide themselves the context information at runtime, whereas the implicit approach uses the device sensors to provide dynamic context information without the need for the user to intervene. The explicit provisioning of contextual information in context-aware recommendation systems has limited the practical incorporation of contextual information in recommendation processes, and has greatly influenced negatively the efficacies of those systems. In the next sections, the elements of the proposed context-aware framework are further described.

3.3 Contextual framework development

Because mobile phone sensing for media recommendations is still in its infancy, there is yet a consensus or standard for mobile phone-based context sensing architecture [161]. For example, there is no standard on how to collect data from mobile phone inbuilt sensors, on how to process and infer additional meaning from the collected data, on how to represent, and communicate the inferred contexts. Although there have been many proposals on context-awareness using mobile devices, however, there is yet a standardized software platform to provide context sensing capabilities for content recommendations on myriad smartphones being shipped daily [161]-[162]. In this section, we introduce a conceptual contextual framework for context-aware recommendations, which uses mobile phone sensing. Figure 3.2 illustrates the high-level layered components of the proposed architecture. Each part is designed to model and to implement certain mechanisms that realize its functionality.

The rationale for the proposed architecture is that present context-aware recommendation applications are expected to interact with diverse types of mobile phone's inbuilt sensors in real time, analyzing and validating sensor information, while associating this sensor information with user's implicitly perceived high-level contextual situations. A truly context-aware personalized recommendation system is expected to reason about this information to provide contextual situations of the user for seamless media recommendations in diverse situations on diverse devices.

Consider this archetypical scenario involving context-aware recommendations. *Closer to her home on a Friday at 8:30 PM, Ana is doing her weekly jogging in the park nearby. Normally, at the end of a working week, using her smartphone, Ana enjoys watching video clips with previews of the latest movies playing in local theaters. She relies on the personalized multimedia recommendation service to provide her favorite choices from which she selects at least one interesting movie to watch during a dry weekend with her friends at the cinema. She expects that her selection will match the interests of her friends.*

Contextual information describing the situation of Ana in this scenario can be identified, using the proposed conceptual context framework. For example, *day of the week*, *time of the day*, *location*, Ana's *proximity to her home*, Ana's activity such as *jogging*, *sitting*, *standing*, *walking*, etc. can be dynamically recognized.

Therefore, the main goal of the framework as earlier stated is to provide contextual information required by context-aware personalized recommendation systems to provide quality and relevant suggestions to mobile users such as Ana in the scenario above. The details of how the proposed context framework can recognize this contextual information from mobile device's embedded sensors are discussed in the next sections of the chapter. First, we describe the framework's three main parts in sections 3.3.1 - 3.3.3.

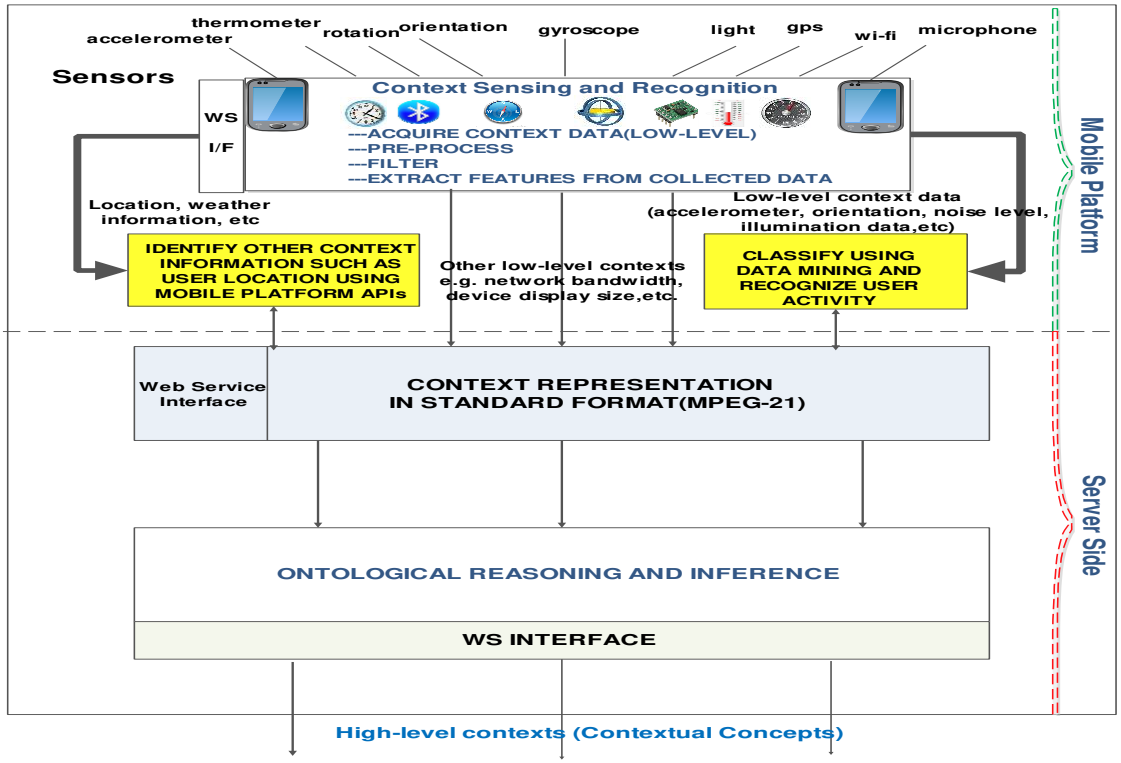


Figure 3.2- Detailed architectural view of the proposed context-awareness framework

The context sensing and recognition provide the sensory services from which high-level context information can be obtained from low-level sensor data. At this level, there are diverse sensors, which we categorized into two classes. First, we have the device inbuilt hardware sensors. Examples of such sensors are accelerometers and gyroscopes, motion sensors used for sensing movement such as acceleration, velocity etc. of the user. For example, GPS sensor provides locations as coordinates, such as *+41.687064 -74.049635*, which do not provide us with any meaningful knowledge. However, a lot of information can be derived from these two numbers, namely house number, street address, locality, city, country etc. Also, take these x, y, z readings, *-0.76615, 1.49398, 8.96389*, of a tri-axial accelerometer obtained from Samsung Galaxy S smartphone, these readings provide no clue that could lead us to knowing that the user of the device was jogging or walking. Essentially, low-level data obtained sensors are aggregated, and extracted as useful data features to be fed into classifies to derive meaningful contextual information.

Second, we have the logical or software sensors, which are available as application programming interfaces (APIs) that can be accessed by context-aware mobile applications via web service interfaces. Good examples are the weather services such as Yahoo Weather and Google Weather APIs. There are other examples of software sensors that can automatically provide user locations as web services. In fact, most hardware sensor's data can be provided as Web Services, and accessed via web interfaces readily available to context-aware applications running on mobile devices.

A good example in this category is the GeoNames API (www. geonames . org). GeoNames is a location database that provides Web service interface access to over 10 million geographic names of diverse GPS coordinates. Apart from providing capability to obtain context information, such as *user is standing,* from the low-level sensor data, the proposed conceptual framework also provides the capability for tapping into the logical sensors to provide contextual information.

3.3.2 Context representation

The development of a context-aware recommendation framework for the delivery and consumption of media items across diverse networks, devices, and users requires a standardized representation of contextual information and user preferences. Because of the diversity and heterogeneity of mobile phone's embedded sensors and the data they provide, there is need to represent context information in a standardized form to address this problem. MPEG-21 (ISO/IEC) provides one of the most complete standards, incorporating standardized tools in the form of XML schemas for the representation and communication of contextual information [20] [126]. MPEG-21 Digital Item Adaptation layer's Usage Environment Description tools (UEDs) provide this functionality. MPEG-21 through the UEDs provides standard descriptions of terminal capabilities, network characteristics, user characteristics, and natural environment characteristics. Additionally, it provides schemas for standardized descriptions of user characteristics and preferences, e.g. user preferences for specific content genres.

3.3.3 Ontological reasoning and inference

The data collected and classified at the context recognition layer can be inconsistent, i.e. the contextual knowledge often contains contradictory facts. Additionally, this contextual information often does not provide an adequate, meaningful contextual situation in which user prefers to consume media items. Ontology based context model uses a description of context concepts and relationships to represent and relate contextual information to obtain more meaningful contextual information in human understandable form [129], [163]-[164]. In order to ensure consistency of contextual data, and to derive more meaning from them, there is a need to incorporate an ontology based model, which takes as input context information in the form of MPEG-21 UEDs and produces as outputs high-level contextual information. Ontology based context model has been adopted and used extensively to provide human and machine understandable contextual information. Ontology is a formal representation of knowledge, such as contextual knowledge [129]. It uses formal languages, such as Web Ontology Language (OWL) with a rich set of operators that allows the use of reasoners to construct contextual knowledge from atomic contextual information produced by the context recognizer layer. The reasoner can check whether the definitions and statements in the ontology model are mutually consistent. It also ensures that each concept fits under its expected definition. An ontology model consists of classes, properties, individuals and restrictions. The restriction ensures that individuals in the ontology model meet specified facts for them to belong to any class. In this way, the model can ensure the consistency of the contextual knowledge.

To infer new knowledge from the contextual information using the ontology model, semantic web rule language (SWRL) [165] built on OWL is used to define rules on individuals and classes. A SWRL rule contains an antecedent part and a consequent part, both consisting of positive conjunctions of atoms. SWRL also has arithmetic operators that can be used to compute desired behaviors of user's dynamic contexts from multiple components. Because of its natural support for superior reasoning capabilities, an expressive formalism for representing complex contextual knowledge, its ability for sharing of knowledge, its ability for automatic derivation of additional knowledge from basic contexts and its ability to detect inconsistent context information to avoid erroneous context inference, this layer of the architecture proposes the use of ontology based context model with SWRL to provide high-level contextual information to the recommendation framework.

3.3.4 Contextual knowledge base

This is a repository where contextual information and other data pertaining to the generation of dynamic context information, such as user information and preferences can be stored.

3.3.5 Application layer

The application layer provides the interfaces necessary for context-aware applications to utilize the contextual information provided by the lower layers of the architecture. The interfaces are designed to be implemented as web services, thereby allowing platform-neutral interactions. Thus, if contextual information is provided as Web services, a wide spectrum of context sensitive applications can be developed on top of the framework accessing the contextual information provided by the framework. Given that this is more of an implementation issue, details of this layer are provided in chapter 5, which discusses the design and implementation of a prototype system adopting the concepts of the proposed system.

3.4 Mobile device context monitoring and recognition layer

This layer of the context-provisioning framework is responsible for the acquisition and recognition of contextual information from mobile device built-in sensors. In this section, the thesis presents the use of smart device built-in sensors to collect low-level contextual data, and using data mining algorithms to infer meaningful high-level contextual information from the collected low-level data [166]. The goal is to take advantage of mass-marketed smartphone built-in sensors, particularly those sensors not explored by previous works, to identify important user daily activities and corresponding locations as well as environmental conditions. This information can be used to build several interesting ubiquitous applications such as personalized multimedia applications that usually require dynamic capturing of mobile user's contextual information without interfering with the user's daily routines [111]. Such applications require that the users' content consumption history and corresponding contexts be registered while conducting their daily routines. Please note that we loosely define contexts as the information that characterizes user consumption. Thus, context information includes user location such as home, office, etc., activities such as walking, sitting, driving, jogging, etc., which are performed by the users when consuming content taking into account the environment conditions such as noise level and illumination.

3.4.1 Context recognition methodology

In a mobile environment, contextual information influences user's media consumption preferences, including the selection and the delivery of the media items [38]. In the proposed framework, context is characterized by several attributes. The most important, like any other context-aware system, is the location of the user. However, location in CAMR is characterized by other contexts such as the noise level, illumination, presence of other persons (company) in the location, the contexts of the user in that location, and time (day of the week, and time of the day). In this section, we describe a simplified model (as illustrated in Figure 3.3) for high-level context recognition from low-level smart device built-in sensor data. The recognition process consists of three successive sequences of phases. (1) The low-level mobile device built-in sensor monitoring and acquisition. This involves logging the raw data from the accelerometer, orientation sensor, rotation vector sensor, GPS, CellID, proximity sensor, battery, light sensor, etc.

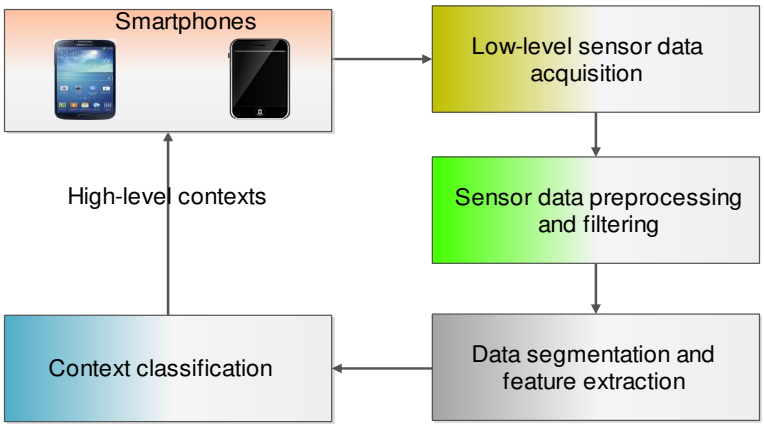


Figure 3.3- High-level view of context recognition process

(2) Preprocessing the logged sensor data. (3) Feature extraction. (4) Analysis and classification of basic contexts, and identifying the overall user contexts. First, we introduce some important concepts necessary to understand the workings of the model, especially the sensors from which it acquires low-level data. Figure 3.3 provides a high-level view of the context recognition algorithm whose functions are elaborated in this section.

3.4.1.1 Mobile phone built-in sensors for data collection

In this section, we discuss mobile phones sensors in the proposed framework and contextual information obtainable from their low-level data.

3.4.1.1.1 Smartphone sensors

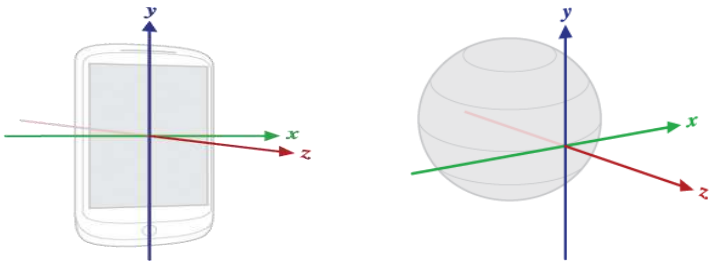
Modern mobile phones as illustrated in Figure 3.4 offer a wide range of built-in sensors. Similarly, they provide various application-programming interfaces to access the data provided by these sensors for low-level context data acquisition. They possess natural characteristics that enable context sensing [103]-[104]. In addition, they come with relatively high processing power and sufficient memory for data processing tasks [38]. They also provide adequate storage space for raw and computed data. They are better as natural, unobtrusive devices for context sensing because they are likely to be with users during their important daily routines [97]. With these characteristics, smartphones have become more practical context sensing devices compared to wearable sensors. However, continuous sensing, monitoring and acquisition of data from a mobile phone's sensors can be very expensive in terms of power consumption, and users are sensitive to the issue of battery drain. Although, acquiring sensory data from one sensor can provide adequate information to determine a user's specific context, however, certain contextual information requires a combination of data from multiple sensors for accurate recognition. For example, to determine the activity of a user at a given location and time requires data from GPS sensors, motion sensors such as accelerometers, gyroscopes, etc. Nevertheless, GPS is a power hungry sensor, especially when it receives data from multiple satellites. Therefore, a conceptual framework for context recognition must consider this factor and provide means to minimizing power consumption. The following section introduces the smartphone sensors adopted in our investigation and the types of high-level contexts that can be obtained using them.

Figure 3.4- Various embedded sensors in modern phone - e.g. Galaxy S4 phone representative of numerous sophisticated mobile devices with a growing number of embedded sensors

(i) Accelerometer sensor*:* Generally, the accelerometer sensor measures the acceleration, in m/s^2, along three axes (x, y, z, see Figure 3.5 a) relative to free fall [112]. It outputs the acceleration of the device in those three axes, by measuring consequent forces applied to the device. The force of gravity has always influenced this measure. Researchers of activity context recognition commonly use the accelerometer for motion based context recognition

[97], [100]-[104], [107], [129], [114], [148], [166]. However, accelerometer data can be combined with data from other sensors such as orientation and rotation to produce better recognition accuracy.

(ii) Rotation vector sensor: This synthetic sensor computes the rotation of the global coordinate system with respect to the device's coordinate system, using accelerometer and gyroscope [112]. The rotation sensor represents the movement of the device as a combination of an angle θ and x, y, z-axes, in which the device has rotated. The three components of the rotation vector are x*sin (θ/2), y*sin (θ/2) and z*sin (θ/2) in x, y, and z-axes respectively [189]. The rotation vector sensor can be used to discriminate between contexts that are very similar, such as running and walking. No other work, to the best of our knowledge, has investigated this possibility [38].



a b

Figure 3.5- (a) Coordinate system relative to a device used by Android accelerometer sensor API and (b) the coordinate system used by the Android rotation vector sensor API [189]

(iii) Orientation sensor: The orientation sensor computes the phone's position and orientation by processing raw data from the fusion of geomagnetic and accelerometer sensors using the Kalman filter algorithm [112]. The geomagnetic sensor measures the strength of the ambient magnetic field in micro tesla (uT) in the x (Azimuth), y (Pitch) and z (Roll) axes. The geomagnetic sensor and the accelerometer data are passed to a function that produces the orientation of the device [112]. The resulting orientation data can be used to measure the physical positioning of the device relative to the earth frame of reference. Examples of such device's positions are portrait up, portrait down, landscape left, landscape right, face-up, and face-down. A very interesting feature of this sensor is its high sensitivity and responsiveness, which are useful in tracking dynamic changes in user's contexts based on the smartphone orientation.

(iv) GPS: Global Positioning System consists of a group of satellites orbiting the earth and receivers such as smart devices, and control stations that monitor and adjust the GPS. The receivers embedded in smart devices are useless without the rest of the GPS system [112]. The receivers use the information being constantly transmitted by the GPS satellites while orbiting the earth to calculate its current location. The satellites follow a defined path, and a receiver must be in "*line of sight*" with at least four of the GPS satellites to receive the information being transmitted by the GPS for location computation. This is one of the major drawbacks for using GPS as location sensor [112]. For example, when the user is indoors, it is difficult for the device GPS receiver to obtain location information from the satellites because of physical obstructions (being out of sight) such as the concrete building. Another disadvantage of GPS is that it is a power hungry sensor; therefore, continuous use of GPS drains the device battery. Thus in the proposed context-awareness framework, the GPS received is probed at intervals to determine if the location of the user has changed. In addition, the location sensing is alternated between GPS and WiFi based location sensing especially when the user is in indoor locations.

(v) Network based location sensors: Other sources of location information apart from GPS are wireless and mobile networks. The location information can be obtained from wireless network sources such as wireless access points and

cell towers. User's locations can be sensed by tracking nearby WiFi access points and their corresponding signal strengths. The collected WiFi information includes the media access control address (MAC) of the access points.

(vi) Proximity sensor*:* Proximity sensor detects the presence of nearby or target objects without any physical contact. These sensors come in various forms, namely, inductive proximity sensor, ultrasonic proximity sensors, capacitive proximity sensors and photoelectric proximity sensors. The basic function of a proximity sensor, depending on its technology is to produce an electromagnetic or electrostatic field or a beam of electromagnetic radiation to detect changes in the field when the target objects come close to the sensor. These sensors are used in mobile phones to sense the closeness of the phone to certain objects such as accidental touch on the screen when held close to the user's ears during phone calls. The sensor can be used to deactivate the touchscreen, for example, if it is detected that the phone is kept in the pocket or wallet. When used together with a microphone, the sensor can enable to detect whether a user is located close to other people.

(vii) Light sensor: A light sensor generates an output signal indicating the intensity of light by measuring the radiant energy that exists in a narrow range of frequencies, and which ranges from infrared, through visible light up to ultraviolet light spectrum. Most mobile phones available today in the market come with light sensors, with the supporting application-programming interface. Data from light sensors can be used to determine the location illumination or brightness. This information can be used to adapt the brightness of the mobile phone screen while playing rich media items, or even as supplementary data to help inferring high-level contexts such as indoor versus outdoor.

(viii) Microphone: Microphone can be used to recognize contexts such as environmental noise level, by continuously listening and classifying ambient sounds. Microphones are one of the oldest sensors in mobile devices.

3.4.1.1.2 Smartphone recognizable contexts

In this section, we present the context states that can be recognized by the conceptual framework, using the smartphone built-in sensors and additional processing functionality.

(i) User movement and location

The movement of a user can be estimated by using a number of different sensors. GPS and WiFi cell indications have been used traditionally to locate and track mobile users. However, these two types of data may not always be a reliable source of context information. GPS has limited usability indoors or anywhere there is no line-of-sight with the satellites. Additionally, common GPS receivers' precisions are limited from 5 to 10 meters. WiFi information depends obviously on availability of WiFi connection at the place where the user is, which might not always be the case. Still, it has proved to be quite reliable when the user is in fact connected to a WiFi network. To complement the use of GPS and WiFi data, three other sensors can be used in combination, to enable acquisition of information about the movement of the user: accelerometer, gyroscope or rotation and orientation sensors. The proposed framework explores the use of these three sensors and others, monitoring their availability and accordingly switching from one to another. Data collected by these sensors can be used to calculate the current speed and even predict the future position of the user.

(ii) Points of Interests

The user's point of interest (POI) is a high-level form of user location information, which is obtained by the platform using the location data obtained from GPS or WiFi low-level data. This enables the framework to obtain the present nearest POIs or those within the range specified by the user. Software APIs such as Yahoo GeoPlanet and Google Places can be used to obtain and process data about many types of POIs such as bars, restaurants, shops, taxis, etc. The framework also provides an interface where users can manually specify their daily POIs that cannot be obtained from the standard APIs.

(iii) Device

The information of the user device such as its screen size can be used to adapt the presentation of recommendations provided to users. In addition, battery status information namely battery level and charging states can be used to switch the modality of media items being presented or to switch off the system to optimize battery power. For example, if the system is aware that the battery lifetime left is less than 10 minutes, the system could use this information to switch from full video presentation to audio only presentation.

(iv) Network conditions

The framework can obtain network conditions such as available bandwidth by measuring the upload and download rates of the network that the device is connected. For example in the implementation of the proposed context-awareness framework, mobile device platform network APIs, e.g. Android WiFi API was used to obtain network condition information. This information can be processed to adapt the presentation of the media item presentations.

(v) Time

The time at the location of the user can be obtained from the device's clock. From these data, we can obtain time information and process it to obtain high-level time information such as time of the day, and day of the week. This information with the location information could be used to know when, where and what type of media items to recommend to the user.

(vii) Proximity

The proximity data can be used to obtain whether a user is idle or not. It can also be used to know if it is appropriate to recommend media items to users, knowing whether the user is alone or in the presence of other people. For example, if we know that the user is busy with the device, this information can be used to push the recommended media items to him via notifications. The proximity information is obtained in our proposed framework by collecting low-level data from proximity sensors provided with the adopted Android platform.

(viii) User information

The user information includes his/her activities as well as preferences when performing such activities. In addition, optional user information such his/her demographic characteristic, e.g. name, age sex etc., as part of the user information. Specifically, user activity defines the context that represents what the user performs as part of his movement contexts, especially physical activities such as, "*walking*", "*jogging*", "*lying*", "*running*", "*standing*", "*driving*", and "*sitting*". The framework also defines more complex activities, namely by associating location information or POIs to the user movement information. In that case, it was possible to define the following "complex" activities: "*ascending stairs*", "*descending stairs*", "*ascending in elevator*", and "*descending in elevator"*. The activities with location information are "*sitting/standing at home*", "*sitting/standing in office*", "*sitting/standing in a bus*", "*driving a car*", "*sitting/standing in train*", " *ascending in elevator*", and " *descending in an elevator"*. Figure 3.6 illustrates low-level form of the signals obtained from some of the sensors described earlier that can be used to infer user activities.

(ix) Environment

Environment contexts such as illumination, noise level as well as weather information can be automatically obtained from the user device. For example, modern mobile devices comes with light sensor, which can be used to obtain the illumination of the environment. Additionally, noise level can be obtained from the device microphone. There are APIs provided by most mobile platforms, which can be accessed to obtained environment weather information on the fly. For example, the implementation of the context-awareness framework proposed in this thesis, Android provided weather API is accessed to obtained weather information. The APIs uses the low-level GPS data (latitude and longitude) to infer location address using the Google Geocoding API. The zip code or the WOEID (where on earth ID) of the address is then used to retrieve weather information from the Google weather or Yahoo weather APIs.

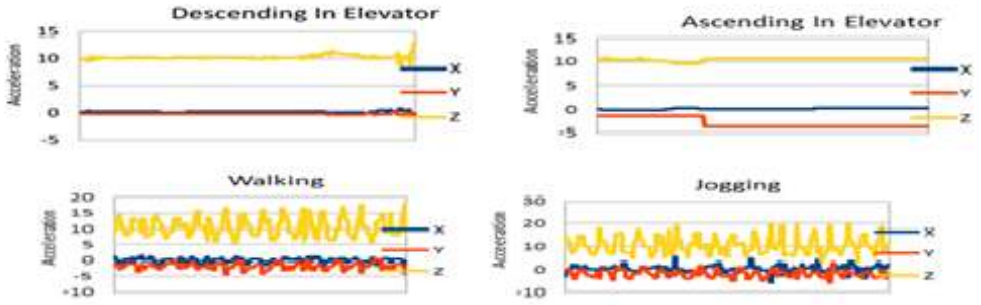


Figure 3.6 - Low-level signals of some user activities obtained from mobile phone embedded sensors

3.4.1.2 Context awareness

Sensors measure specific physical parameters from the environment where they are situated. Context-aware systems may directly use this low-level information to derive semantic or high-level context states or situations. In the previous subsection, we have precisely addressed these aspects by describing available sensors on mobile devices and the data they are able to acquire and the context states that are possible to derive by using the low-level data. In this section, we describe the actual processes required to obtain such data. Context-awareness deals with the ability of the proposed framework to make sense of the data collected automatically by the mobile device's built-in sensors. This is accomplished by executing a number of steps, namely context sensing, context filtering, feature extraction, context modeling, training and classification. The remainder of this subsection describes the first three, whereas following subsection is dedicated to the latter.

(a) Context sensing

In the proposed conceptual context awareness framework for personalized recommendations, context sensing is defined as the acquisition of low-level context information in the user's environment, via mobile devices to capture the real-world situations to offer personalized recommendations. These data from embedded mobile device sensors are error prone, additionally, at their low-level form, they are practically difficult to use for building real-world applications. Noise, drifts, and delays are some of the common sources of sensor data errors [112]. In order to reduce the influence of these sources of errors in the sensed data, since they can corrupt the captured context information, filtering the sensor signal is an important process to perform. This is necessary because applications utilizing the sensor information have no control over the outputs of the sensor but can process the output to reduce the errors.

(b) Filtering

In our framework, we focus on the use of the three mentioned user activity sensors (accelerometer, gyroscope/rotation and orientation) to design an efficient approach to recognize user activities. Since raw data from mobile device's in-built motion and position sensors, such as accelerometers and orientation sensors respectively, are highly prone to high frequency noise and errors, it is important to eliminate or minimize the influence of these factors on the emitted signals before using them to recognize context information. The signals from these sensors are streamed through a low pass filter to filter out the high frequency thereby smoothing the signals. The basic principle is to simply replace the values of a sample by the moving average calculated as the set of $2N + 1$ values, made of the current sample values and the N values preceding it. This type of filtering is called a moving average filtering because it smoothens the

sensor data by replacing each data point with an average of neighboring data points within a given order of the filter [167]. This low pass filter process produces a response given as the difference equations 3.1 for a tri-axial sensor. For a one- axis sensor such as a light sensor, the filtering is performed on the streams of data it generates.

$$\begin{cases} x_{s(i)} = \frac{1}{2N+1}(x(i+N) + x(i+N-1)+, \dots, x(i-N)) \\ y_{s(i)} = \frac{1}{2N+1}(y(i+N) + z(i+N-1)+, \dots, y(i-N)) \\ z_{s(i)} = \frac{1}{2N+1}(z(i+N) + z(i+N-1)+, \dots, z(i-N)) \end{cases} \quad (3.1)$$

$X_s\ (i)$ is the filtered signal value for the i^{th} data point; whereas N is the number of data points on both sides of $X_s\ (i)$ and 2N +1 is the order of the filter. There are two important advantages for using this method to filter out noise from motion sensors according to [167]. First, while retaining low frequencies, it minimizes random high frequencies in the sensor data. Second, it helps to reduce errors that might have been introduced into the data while collecting data from the sensors.

(c) Feature extraction

Feature extraction is a signal processing technique that is used commonly to extract useful hidden information from raw data, transforming the entire raw data into a useful and reduced representation set of features [107]. Though raw sensor data from smartphone contain lots of hidden information and noise [168], the feature extraction process, if carefully selected, can help to isolate useful features from unwanted ones. In addition, having redundant features in a large set of features can increase computational requirements of the classification algorithms as well as jeopardizing their recognition accuracies. Therefore, extracting suitable features of smartphone sensor data is a very crucial process, which can improve the whole classification efficiency. The feature extraction process in our investigation is done in two phases. The first one corresponds to the process of splitting the acquired data into fixed length segments or windows; the second one is the actual process of extracting relevant features from each defined window.

(i) Splitting phase

In the first phase, a process known as fixed length temporal sliding window algorithm is used [114]. This algorithm splits the sensor data into data segments of fixed intervals of samples called "windows". A window contains a small part of the sensor signal [107], [114]. Each window is "overlapped" or "*slided*" to form the next window, preserving a proportion of the previously sampled signal data to be used as the beginning of the next sample [103]-[105], [111]. The formal definition of this process is provided below.

Let f represent the function that determines the patterns in the event sequences of N smartphone built-in sensors with L axes (x, y, z), matching the user contexts. f is defined as follows:

$f : S^l \rightarrow A$ where l is the window length of each sensor's axis and A is a set of contexts to be recognized.

Let $s \in S = \{ s_1, s_2, \dots, s_n\}$, and let $a \in A = \{a_1, a_2, \dots, a_n\}$ be defined as the representations of the smartphone built-in sensors and contexts to recognize respectively.

Each sequence of events of a sensor s is represented as vector X^s, Y^s and Z^s defined as:

$$\begin{cases} X^s =< x_0^s, \dots, x_i^s, \dots > \\ Y^s =< y_0^s, \dots, y_i^s, \dots > \\ Z^s =< z_0^s, \dots, z_i^s, \dots > \end{cases} \quad (3.2)$$

representing the readings of x, y and z axes' events respectively at time i.

A function f_1 take as input $N \times L$ sequences of sensor events to produce as output K vectors of features Fi.
Each vector is labeled with activity a ∈ A = $\{a_1, a_2, \ldots, a_n\}$. Let *l and i* denote windows parameters, representing the window's length and the timestamp when the first window begins respectively. $i+l$ is the total time for the first window and it marks the time when the next window begins. Let r be the length of the windows slide. For a 50% windows slide, $r = 0.5\ l$.

Using these definitions, function f_1 is defined as follows:

$$W_i : X^s_i \rightarrow M_i, \quad f_1 : M_i \rightarrow F_i \tag{3.3}$$

Where $M_i = <M_{xi}, M_{yi}, M_{zi}>$ are matrices of temporal groups of sequences of events for each sensor in x, y, and z axes.

And W_i is the sequence of events segmented into d samples of time domain windows or time slices of l seconds in length for contiguous readings of the sensor's x, y, z axes respectively, starting at the time i as follows:

$$\begin{cases} w^s_{xi} = < x^s_i, \ldots, x^s_{i+l-1}, \ldots > \\ w^s_{yi} = < y^s_i, \ldots, y^s_{i+l-1}, \ldots > \\ w^s_{zi} = < z^s_i, \ldots, z^s_{i+l-1}, \ldots > \end{cases} \tag{3.4}$$

The window slide r defines the next temporal windows as follows:

$$\begin{cases} w^s_{xi+r} = < x^s_{i+r}, \ldots, x^s_{i+rl-1}, \ldots > \\ w^s_{yi+r} = < y^s_{i+r}, \ldots, y^s_{i+rl-1}, \ldots > \\ w^s_{zi+r} = < z^s_{i+r}, \ldots, z^s_{i+rl-1}, \ldots > \end{cases} \tag{3.5}$$

For each window, the segments that start at time i are grouped into the matrix:

$$\begin{cases} M_{xi} = < w_{xl}, \ldots, w^N_{xl}, \ldots > \\ M_{yi} = < w_{yl}, \ldots, w^N_{yl}, \ldots > \\ M_{zi} = < w_{zl}, \ldots, w^N_{zl}, \ldots > \end{cases} \tag{3.6}$$

(ii) Feature extraction phase

The second process involves generating a set of statistical features from each window, a process popularly known as feature extraction [107-108]. Following the sliding windowing process explained in the previous section, these statistical features are generated from each of the matrices M_{xi}, M_{yi}, M_{zi} to build vector F_i with labeled contexts $a \in A = \{a_1, a_2, \ldots, a_n\}$, which the user performs. Therefore, given a set of n activities $A = \{a_1, a_2, \ldots, a_n\}$, every temporal window w_i will produce a vector F_i that is labeled with activity $a \in A$. A function f_2 builds the classifier that learns and finds the mapping between F_i and the activity A performed by the user.

To minimize computational cost of the classifiers, we have adopted the time domain statistical features because they have been shown to deliver good results with minimal computational costs in the previous works [97], [103],[107]. Time domain features are computed on the fly with minimal memory and computational costs, compared with their frequency domain counterparts such as Fast Fourier Transform (FFT) [97]. The features are extracted with overlapping at 50%. The 50% overlapping is chosen because it has been reported in the existing works as an adequate value for the context classification process [104], [118], [169]. To determine suitable window length l, with a predetermined number of samples, we have investigated the extraction of features using window lengths with $l = 64, 128, 256, 512$, and *1024* samples as reported in [38]. Ninety (90) features were extracted from the combination of orientation, acceleration, and rotation vector sensors (all are tri-axis sensors). Additionally, 30 separate features were extracted from each sensor (10 from each axis). Table 3.1 presents summarized definitions of various statistical features explored in the classification process. This way, the classifiers can be encoded with these sliding data windows, corresponding to the high-level context information for recognizing user's future contexts. *Note*: In the definition of *Median*, $\sum f_0$ is the sum of the frequencies or number of scores up to the interval containing the median. l is the lower

limit of the interval containing the median. N is the total number of values in the series. f_w is the frequency or number of series within the interval containing the median and i is the size or range of the interval.

Table 3.1- Feature descriptions and definitions

Feature	Feature Description	Formula
Variance (Var)	Defines the average squared difference from the mean of the device's acceleration, rotation and orientation in 3 directions over a sliding window.	$var = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)$
Maximum (Max)	Maximum is the largest value of a sequence of discrete acceleration, rotation, and orientation of the device in 3 directions over a sliding window.	$f(x_i^*) \geq f(x_i)$
Minimum $f(x^*)$	Minimum is the smallest acceleration, rotation and orientation event in 3 directions of the device over a sliding window.	$f(x_i^*) \leq f(x_i)$
Median (Med)	Median is the value that separates the higher half from the lower half of acceleration, rotation and orientation events of the device in 3 directions over a sliding window.	$median = l + \left[\frac{\frac{N}{2} - \sum f_0}{f_w}\right] i$
Mean	Defines the average acceleration, orientation and rotation of the device in 3 directions over a sliding window. It smoothens the sensor's signals in its 3 directions.	$mean(\mu) = \frac{1}{n}\sum_{i=1}^{n} x_i$
Standard Deviation (STD)	Measures the distribution of the device's acceleration, orientation, and rotation in 3 directions over a sliding window.	$STD(\delta) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$
Root Mean Square (RMS)	It represents n discreet acceleration, orientation and rotation of the device in 3 directions over a sliding window.	$RMS = \sqrt{\frac{x_1^2 + x_2^2, \ldots, x_n^2}{n}}$
Zero Crossing (ZC)	This represents the points where acceleration, orientation and rotation of the device change from negative to positive or vice versa in the 3 directions over a sliding window.	$Zcr = \frac{i}{X-1}\sum_{x=1}^{X-1}\{X_t X_{t-1} < 0\}$
Sum of Squares (SOS)	Determines the variability of the distribution of acceleration, orientation, and rotation values in 3 directions over a sliding window.	$SOS = \sum_{i=1}^{n}(x_i - \bar{x_i})^2$
Range	This is the difference between the largest and the smallest values of device acceleration, rotations and orientation in 3 directions over a sliding window. It provides the statistical dispersions of the sensor readings.	Range = max-min

(iii) Feature selection phase

Because of the large feature space in the last phase, using the entire extracted feature set might be computationally expensive due to the high dimensionality problem. Additionally, it is also possible that not all the extracted feature would contribute to the discrimination property of the recognition model. Sometimes, some can introduce noise thereby reducing the recognition accuracy. Therefore, there is the need to check the extracted features for relevance and redundancy in a process called feature selection in order to select the best feature for context classification. Two characteristics of each feature are determined. The relevance of a feature defines its ability to discriminate context classes, whereas, the redundancy determines the similarities between these features. If two features are similar, then, one of them is dropped. This process can be preceded by discretization process depending on whether the data require conversion from numeric or nominal or categorical formats. After obtaining the relevance and redundancy of the feature set, the data can be processed by the search algorithm to select the features that meet the relevancy and redundancy requirements.

3.4.2 Context model building, training, and classification

3.4.2.1 Building context recognition models

The interpretation of sensor's low-level data to realize the move from low-level to high-level context information runs in steps. The statistical features generated in the last process are fed into the classifiers to produce the context predictive models. These models can then be ported on the mobile phone to infer meaningful contextual information.

The classifiers decide the feature space to which the unknown sensory patterns belong. The classification process is done by using the feature vectors obtained from the data windows or segments as inputs to the classification algorithms, which associate each window with the corresponding high-level context class [205].

To realize this process, first, each model was trained and its performance was evaluated using the cross validation approach. Second, separate datasets can be used for training the models and for evaluating them. This is designed to determine the generalized performance of each algorithm to ascertain the ones with the best performance for building the context prediction model. To select and to build the appropriate predictive models, we have evaluated a number of classification algorithms, with the aim of identifying the algorithms that best recognize each context being investigated. The algorithms include well-known base-level classifiers that have been used in existing context and activity recognition projects [168]-[170]. Next, we introduce some classification algorithms that could be used to model context recognition. These algorithms are k-Nearest Neighbor (kNN), kStar, Random Forest (RF), C4.5, Naïve Bayes, Support Vector Machine (SVM), Multilayer Perceptron(MLP), Logistic Regression (LR), RIPPER: Repeated Incremental Pruning to Produce Error Reduction, SMO, and PART: Projective Adaptive Resonance Theory. In chapter 6, we summarize and compare the performance of popular algorithms used in context recognition [See Table 6.1 in chapter six]. The goal here is not the development of new classification algorithms but to use these algorithms to model context recognition and incorporate it in the context framework. We present the evaluation of these predictive models in chapter six. For details on these algorithms, readers are referred to [166], [168]-[170], however a brief introduction of these algorithms is presented in the next section.

3.4.2.2 Classification algorithms for context recognition

In order to build the predictive models, we have selected a number of classification algorithms, with the aim of identifying those that best recognize each activity being investigated. The algorithms include well-known base level classifiers that have been used in existing context and activity recognition projects. Here we give a brief introduction of the algorithms; readers interested in their full descriptions can consult [166]-[172].

(a) K-Nearest Neighbor: kNN belongs to a group of classification algorithms called instance based learners. The classification of an unknown instance is based on finding the k-closest labeled examples from the training data using some distance measures such as Euclidean distance formula. The majority class of the *k* closest neighbors found is then assigned to the unknown instance. One important feature of this algorithm is that most of the processing is done during testing rather than during training. Therefore, these algorithms are very fast with negligible training and testing computational time. In addition, their complexity is independent of the number of classes for which algorithms that are more complex exhibit long processing time and high computational costs.

(b) kStar*:* Like kNN, kStar is an instance-based learner. However, unlike *kNN*, it uses an entropy distance measure based on probability of transforming an instance into another instance by randomly choosing between all possible transformations.

(c) Decision Trees: are a class of supervised learning techniques that are generally defined as trees whose internal nodes are tests(on input patterns) and whose leaf nodes are categories of patterns. A decision tree algorithm assigns a class number of an input pattern to an output by filtering the patterns down all nodes in the tree, with each node having mutually exclusive and exhaustive outcomes. A good example of decision-tree building algorithm is C4.5 [172]. Random forest [171] is a decision algorithm based on decision trees.

(d) Naïve Bayes: is a classification technique based on the so-called Bayesian theorem and it is particularly suitable when the dimensionality of the input data is high. In order to apply this algorithm to analyze data, the instances to be classified need to be represented by numerical (discrete) features. It, however, assumes that all features of a class are considered independent. During learning process, a set of feature vectors is fed into the algorithm, each vector labeled with the class the instance represents and or/belongs to. Thus, the combination of features with high probability can be determined. Given this information, during application, one can easily compute the probability of a new instance either belonging to a class or not.

(e) Support Vector Machine (SVM): provides a set of related supervised machine learning algorithms that views input data as sets of vectors in a *n*-dimensional space by constructing a separating hyperplane in that space. The separating hyperplane maximizes the margin between the two data sets.

A good separation is normally achieved by the hyperplane that has the largest distance to the neighboring data points in both classes. The larger the margin, the better the generalization error of the classification process. Two common implementations of SVM are Sequential Minimal Organization (SMO) and LibSVM [118].

(f) Multilayer Perceptron (MLP): is a feed forward artificial neural network model that maps sets of input data to a set of appropriate outputs. MLP consists of multiple layers of perceptrons; at the bottom level are the features from the inputs and at the top level are the outputs that give estimation for the class. Each perceptron (node) computes a single output from multiple inputs by forming a linear combination according to their input weights, producing an output through a nonlinear activation function. The main advantage of MLP is that it can distinguish data that are not separable by linear techniques. MLP utilizes a supervised learning technique called back propagation for training the network.

(g) Logistic Regression: is a regression technique for analyzing data with categorical dependent variables. It determines the impact of multiple independent variables presented simultaneously to predict memberships of one or other of the two dependent variable categories. This classification algorithm, in addition to predicting group members, can also provide us with knowledge of the relationship and strength among variables [118].

(h) RIPPER: Repeated Incremental Pruning to Produce Error Reduction is an inductive rule learning technique that generates a set of rules to identify classes while minimizing the classification error. The number of training examples misclassified by the rules defines this error. It consists of outer and inner loops. The former is used to add one rule at a time to the rule base while the latter is used to add one condition at a time to the current rule. One defining characteristic of this classifier is that no a priori assumptions are made regarding the final class. The inductive learning algorithm, however, works on a primary assumption that the trained data are similar in some way to the unseen data. One good advantage of this learning technique is that it can be applied to large noisy dataset.

(i) PART: Projective Adaptive Resonance Theory is an inductive rule based learner, similar to RIPPER that uses divide-and-conquer iterations to build a partial decision tree based on C4.5 in each iteration and converting the best leaf into a rule.

3.4.2.3 Mobile context data acquisition application (CDAA)

All the processes and concepts required to build automatic context recognition on devices as presented in this section were developed into a context recognition application with which various classification models could be evaluated for incorporation into the context framework. This section describes the mobile context collection process. Similarly, it describes the mobile application developed for the automatic acquisition of user's contextual information based on those processes.

In the literature, contextual data are usually collected using three major techniques [101], [169]. The first technique involves collecting context data using wearable sensors attached to the bodies of users in controlled experimental settings, whereas the second involves using device built-in sensors. The wearable sensor approach has some drawbacks. Wiring sensors to the bodies of the users are not suitable for practical purposes because they are not convenient, requiring frequent user interventions. They also come with additional cost [104], [112], [116]. On the other hand, the mobile device approach is less costly because no additional equipment is required, with minimal user interventions, and having the potential to reach billions of users. The third method is the manual acquisition of context data by asking users to supply such data themselves or sometimes some try to simulate contextual data. A context-aware system, therefore, should as much as possible be able to recognize, in real time, user's contextual situations with minimal user's involvement. Based on the requirements of the proposed framework, the first and the third methods are not considered. Therefore, CDAA application was developed to build an automatic context acquisition,

processing, classification and recognition model. Figure 3.7 depicts the simplified processes of the CDAA application, and the following sections elaborate more on these processes.

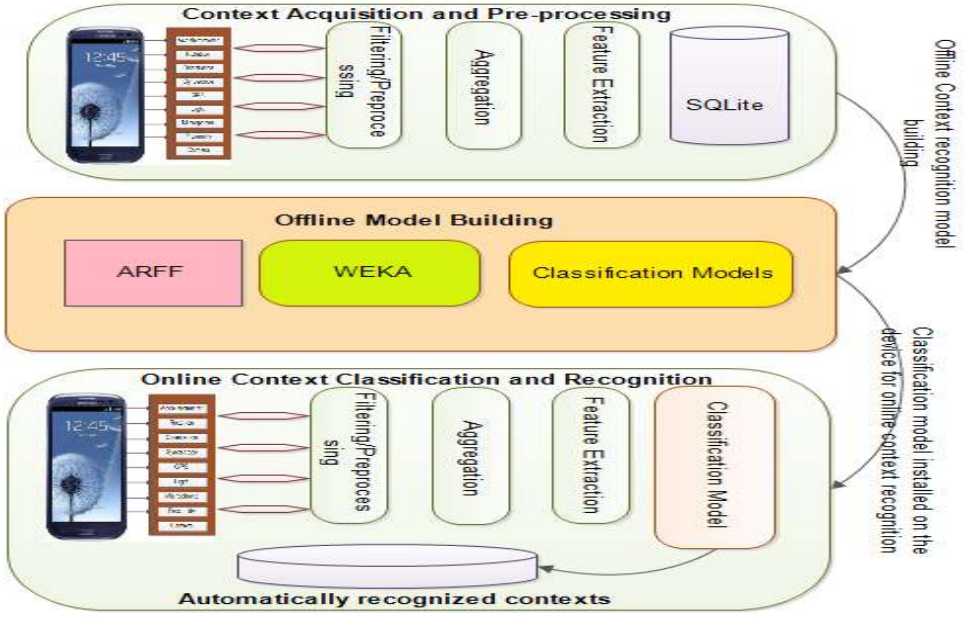


Figure 3.7 - Context data acquisition application (CDAA)

The building of the model encompasses two phases. The first phase involves collection of context data using the mobile application. In the context collection process, the user selects the activity to perform from a set provided in the application interface. The application then collects the corresponding low-level data from the smartphone sensors and labelled sensor data. The collected context data is then used in an offline process to build the classification model. In this process, WEKA framework [172] was adopted. WEKA framework provides a collection of machine learning models implemented in Java language for data mining purposes [173]. These machine-learning models can be used directly, receiving the preprocessed context data features as input. The models can also be utilized from Java applications.

Therefore, in this offline process, the collected data are converted to WEKA file formats (ARFF). Then, the classification models are built based on those data. In the online process, the context recognition models are ported on the mobile device for online recognition of mobile user's contexts in the third process.

3.5 Context representation and ontological model

To effectively aggregate and communicate retrieved contextual information obtained in the last process (context recognition phase) between different components of an external system such as a context-aware recommendation system, context information must be represented in a format that can be understood by each component in the recommendation process. Consequently, the interoperable representation of contextual information is a fundamental precondition for communication between context producers and context consumers. Ontology based context model and MPEG-21 Usage Environment Description (UEDs) representation of the context information are proposed as part of the conceptual framework for building robust context-aware recommendation systems [8]. In this section, the chapter presents the context representation component of the proposed context-awareness framework.

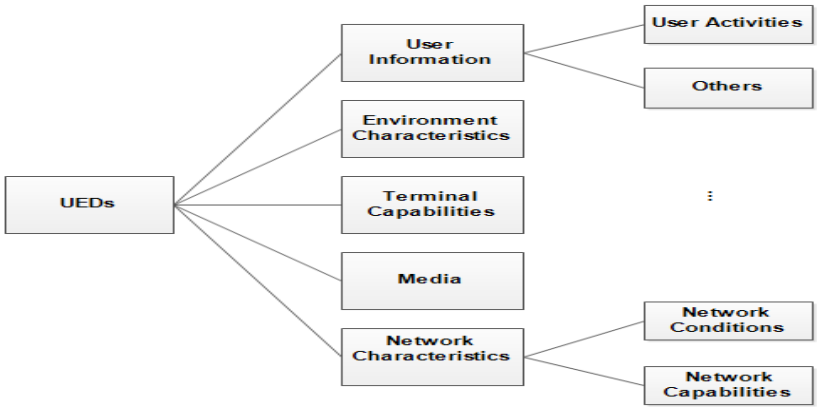


Figure 3.8 - MPEG-21 UEDs concepts

3.5.1 MPEG-7 and MPEG-21 metadata representations

MPEG-7 MDS is used for describing the media item metadata [152]. MPEG-7 semantic descriptions are essential to allow the identification of candidate content and for consequent generation of user and media profiles. MPEG-7 technical characteristics or encoding parameters such as bit rate or spatial and temporal resolution are also essential to tailor the content to the constraints imposed by the context of use namely, terminal capabilities and network conditions. All other contextual metadata are represented using the MPEG-21 tools, namely the MPEG-21 part 7 UED tools [6], [124], [126], [148]. However, these standards were defined to allow the adaptation of content according to the constraints of the consumption environment and user preferences, and not to assist recommendation of the content based on user's contextual situations [6], [146], [143]. The concepts defined in the context representation model based on MPEG-21 UEDs and ontology are illustrated in Figure 3.8 including user activity concept that was added as an extension. These concept are defined in the next subsections.

MPEG-21 (ISO/TEC 21000) has been developed to define a normative open framework for multimedia delivery and consumption for use by all players in the media content delivery chain [6], [124], [126], [139]. The framework has provided media creators, providers, consumers, etc. with equal opportunity in the MPEG-21 media enabled market place. One of the major benefits of representing contextual information in MPEG-21 UEDs for media delivery is that it provides a variety of content consumers with interoperable access to a large and myriad multimedia content [126,147]. It provides the technologies needed to support users to exchange, to access, to distribute, to consume, and to manipulate media contents in an open, efficient and transparent way [127]. Conceptually, MPEG-21 is based on some fundamental components. First, the definition of the fundamental unit of media content distribution known as Digital Item (*DI*). Second, the definition of "*User*", which interacts with *DI*s such as video, audio, and text [174].

(a) MPEG-21 user model

In MPEG-21, "*User*" is any entity (entity can be media creator, producer, consumer, etc.) who interacts with the MPEG-21 environment. MPEG-21 provides the framework where one user can interact with another user via the digital item, also known as the multimedia content.

(b) Digital item

The *digital item*, *DI,* is the basic unit of interaction in the MPEG-21 enabled environment. It is a structured digital object with a standard representation, identification, and metadata. Many entities can be regarded as DIs, e.g., music, news, movies, etc. A piece of news, whether in audio, video or text form is a digital item. Likewise, a movie titled "*Titanic*" is a digital item in whatever formats it is created or represented. This means that there are different types of

digital items, with different representations, possibly in different formats, depending on their contents. Therefore, MPEG-21 provides the required flexibility for interoperable representation of digital items in any applicable format.

(c) Digital item adaptation (DIA)

MPEG-21 comes in a number of standard parts, sixteen of them [160]. However, the most important part of MPEG-21 is part 7, which is the **Digital Item Adaptation** (DIA). The DIA defines descriptive tools for usage environment format features that might influence transparent access to digital items, notably, terminals, network, users, and the natural environment users and terminals are collocated. DIA specifies the descriptive tools for the transparent and interoperable delivery and adaptation of distributed digital items. DIA consists of two important parts, the adaptation engine and the usage environment description (UED).

(d) Adaptation engine

To satisfy the constraints in the digital item delivery environments, DIA provides non-normative tools that specify the constraints and how they can be satisfied in order to deliver multimedia content in a way that guarantees richer user experience quality. The adaptation engine is classified into two types. The resource adaptation engine and the description adaptation engine [6], [160]. The adaptation engines are non-normative because they do not specify how adaptation mechanisms are implemented, but specify the syntax and semantics required assisting the adaptation process.

(e) Usage environment descriptions (UED)

UED standardizes the description and structures of user environment information. Even though UED does not directly participate in the adaptation process, however, it is required as part of the DIA's specification to enable other tools to perform adaptation using the descriptive information. The usage environment description includes information about the terminal capability, network characteristics, user characteristics and natural environment characteristics. These descriptions can provide the fundamental input to either the adaptation engine or the recommendation engine and help to enable user independence. In the next section, we introduce each of the descriptions provided by the UEDs, as alternative standardized representation, which are explored by CAMR.

(1) Terminal capability*:* The terminal is a DIA generic term for all devices, regardless of their locations, that participate in the digital item delivery chain. Therefore, a terminal is not only the end user mobile phones, TV set, laptop, etc., but could also include the server, the proxy, and even network nodes. Hence, the capabilities of the terminal are usually described in terms of transmitting and receiving capabilities. It describes the device class, encoding and decoding capabilities, benchmark information, user intervention possibilities, power characteristics, data input/output characteristics and storage capabilities. Example of terminal capability UED is given in Figure 3.9.

.

```
<TerminalCapability xsi:type="DisplaysType">
        <Display id="d0">
         <DisplayCapability xsi:type="DisplayCapabilityType" bitsPerPixel="8"
colorCapable="true" maximumBrightness="1500">
                        <Mode refreshRate="70">
                          <Resolution horizontal="176" vertical="144"/>
                        </Mode>
                        <ScreenSize horizontal="176" vertical="144"/>
                </DisplayCapability>
        </Display>
</TerminalCapability>
```

Figure 3.9- Terminal capability UED

(a) Codec capability*:* specifies the coding format supported by the device, in particular, it specifies the encoding and the decoding capabilities of the device. MPEG-2, MPEG-4 SP, H.263, H.262 AVC, etc., are good examples of

encoding and decoding capabilities. This information is useful for adapting the DI's format to match that of the terminal. For example, if a terminal does not support the MPEG-4 format of a movie video content, the MPEG-4 video format can be transcoded to MPEG-2 supported by the terminal.

(b) Device properties: specify terminal properties such as device storage and power characteristics. The power characteristics description tool, in particular, is very important because it describes the power consumption of the terminal. Example of interesting power characteristic is a battery level remaining or battery time left. To conserve power consumption of the terminal, this characteristic can be used to monitor the battery level and adjust the behavior of the applications accordingly.

(c) Input/output characteristics: Describe the output of the terminal, such as the display characteristics, e.g. display color scheme, frequency, Signal-to-Noise Ratio(SNR), number of channels, etc. and input characteristics such as keyboard, microphone, touchscreen, etc.

(ii) Network characteristics describe the static and dynamic attributes of the network such as the maximum capacity of the network and a bandwidth guaranteed by the network. Example of UED for network characteristics is depicted in Figure 3.10.

(iii) Network capability: describes the available bandwidth, error and delay. The available bandwidths include-minimum, maximum and average available bandwidth of the network.

```
<Description xsi:type="UsageEnvironmentType">
        <UsageEnvironmentProperty xsi:type="NetworksType">
          <Network>
                <NetworkCharacteristic xsi:type="NetworkCapabilityType"
maxCapacity="512000" minGuaranteed="32000" errorCorrection="true" errorDelivery="true">
                </NetworkCharacteristic>
        <NetworkCharacteristic xsi:type="NetworkConditionType" duration="PT330N1000F">
                <AvailableBandwidth maximum="256000" average="128000">
                </AvailableBandwidth>
                <Delay packetTwoWay="330" delayVariation="66"/>
                <Error packetLossRate="0.05">
                </Error>
                </NetworkCharacteristic>
        </Network>
</UsageEnvironmentProperty>
```

Figure 3.10 - Network characteristic UED

```
<UserCharacteristic xsi:type="UserInfoType">
  <UserInfo xsi:type="mpeg7:PersonType">
        <mpeg7:Name>

                <mpeg7:GivenName>Yomi</mpeg7:GivenName>

                <mpeg7:FamilyName>Otebolaku</mpeg7:FamilyName>
        </Name>
    <mpeg7:UserRole>Student</mpeg7:UserRole>
        <mpeg7:UserActivityType type = Motion>
            <mpeg7:UserActiviy>Walking</mpeg7:UserActivity>
    </mpeg7:UserActivityType>
        </UserInfo>
</UserCharacteristic>
```

Figure 3.11 - User information UED

(iv) User Characteristics are the descriptive information about the user in the digital item delivery chain. A user does not refer only to the end user who consumes the digital item, but also to the creator, the publisher, etc., who is involved in the delivery of the digital item. The user's descriptive information can be used to personalize the selection and the delivery of multimedia items to the user. These characteristics include:

(a) ***User information*** describes a person, a group of persons or an organization. User information includes the identity and demographic information about the user. Example of user characteristic is given in Figure 3.11.

(b) ***Presentation preferences*** specify the set of preferences, which are related to the digital items and their associated resources and the way in which they are presented to the user. Rich tools for specifying rendering of audio-visual DIs, which meet the format or the modality preferred by the user, the presentation priority set by the user, and those that direct his focus and attention with respect to audio-visual and textual digital items.

(c) ***User activity*** describes the activity contexts of the user in a given location and at a given time. The user information UEDs does not include schema for this category of user information. However, to provide the description of the user activity, the user activity schema illustrated in Figure 3.12 was created and included in the user characteristic UED schema.

(vi) Location characteristics describe the mobility of the user and his destination. The former provides the concise description of the user's movement over time. It can be used to characterize a user's movement as pedestrian, highway, or vehicle, etc. Destination specifies the intended location of the user described by the geographic coordinates or a conceptual location information as a list of terms.

```
<complexType name = "UserActivity">
  <complextContent>
   <extension base = "UserInfoType">
            <sequence>
              <element name = "UserActivity"
             TypeMpeg7:agentType minOccurs="0"/>
</sequence>
   </extension>
  </complexContent>
<complexType>
```

Figure 3.12 -MPEG-21-user information UED schema extension for user activity

(vi) Natural environment characteristics

These tools specify descriptions of the physical conditions of the user's environment at any given time. For example, the lighting conditions or the noise level of the environment at a given time is an important factor that can be used to tailor the recommendation of multimedia items to users. The environmental noise level can be used to tune the audio volume of the recommended digital items either increasing or decreasing the volume depending on whether the environment is noisy or quiet. If the environment is noisy, the volume can be increased to reflect this situation. Additionally, the illumination of the environment is another characteristic that affects the perceived color of the user's device display, which can be adjusted by tuning the display's contrast.
The time and location of the user's environment are important factors of the user environment descriptions because the time and location that users consume digital items influence their preferences. Example of natural environment UED is depicted in Figure 3.13.

(vii) Accessibility characteristics

These tools specify the descriptions that enable the adaptation of digital items to the user's auditory/visual impairments, for example, audiogram is specified with various frequency thresholds for both left and right ears.

```
<UsageEnvironmentProperty xsi:type="NaturalEnvironmentsType">
                    <NaturalEnvironment>
                        <NaturalEnvironmentCharacteristic xsi:type="LocationType">
                            <Location>
                                <mpeg7:Name>public</mpeg7:Name>
                                <mpeg7:GeographicPosition>
<mpeg7:Point longitude="-8.61" altitude="10.0" latitude="41.150"/>
                                </mpeg7:GeographicPosition>
                                <mpeg7:Region>pt</mpeg7:Region>
                            </Location>
                        </NaturalEnvironmentCharacteristic>
                        <NaturalEnvironmentCharacteristic xsi:type="TimeType">
                            <Time>
                <mpeg7:TimePoint>2008-03-06T15:22:00+01:00</mpeg7:TimePoint>
                            </Time>
                        </NaturalEnvironmentCharacteristic>
                    <NaturalEnvironmentCharacteristic xsi:type="AudioEnvironmentType">
                            <NoiseLevel>80</NoiseLevel>
                        </NaturalEnvironmentCharacteristic>
                <NaturalEnvironmentCharacteristic xsi:type="IlluminationCharacteristicsType">
                            <TypeOfIllumination>
                                <ColorTemperature>159</ColorTemperature>
                            </TypeOfIllumination>
                            <Illuminance>500</Illuminance>
                        </NaturalEnvironmentCharacteristic>
                    </NaturalEnvironment>
                </UsageEnvironmentProperty>
```

Figure 3.13- Natural environment characteristics UED

3.5.2 Ontology model: the knowledge-based approach

Future context-aware personalized recommendation systems must decouple user from the recommendation processes and services, and perform the necessary tasks of providing implicit contextual information for the provisioning of relevant recommendations. To realize this, context-aware recommendation systems require an appropriate context model to represent, to manipulate, and to access high-level contextual information. Because context-aware recommendation systems are distributed in nature, especially with diverse kinds of clients, such as mobile clients, multiple context services interacting with the context-aware services, the problem of exchanging contextual information becomes a bottleneck. Ontologies can provide a solution by identifying relevant concepts, and specifying their syntactic and semantic representations. This way, context data services and clients that do not have consistent terminology can still share data in an interoperable way. Moreover, ontologies enable to define relationships among context data elements, thus enabling to derive additional knowledge from the combination of initial context data.

Ontology is a formal representation of a set of concepts within a domain and the relationships between these concepts, based on their roles [41]-[42], [127], [175]. Among others, the advantage of using ontology to model contextual information is its ability to provide complex but efficient inference mechanisms to deduce high-level contexts from low-level context data [175]. Developed in the artificial intelligence community, ontology can facilitate context knowledge sharing and reuse. Unlike the MPEG-21 UEDs context representation, ontology provides formal vocabulary and descriptions of the semantics of contextual information in terms of concepts and rules [177].
It allows contextual information to be encoded in a way that context-aware applications can process to infer new knowledge from the user's contextual situations through logic-based reasoning.

3.5.2.1 Ontology preliminaries

The realization of any knowledge base system involves providing a precise characterization of the knowledge to be specified as well as defining the reasoning services the system needs to perform on the knowledge i.e. the kind of questions the system must be able to answer when queried. Providing precise characterization of the behavior is the fundamental principle of a knowledge representation system [96]. It gives a precise specification of the functionality

to be provided by the knowledge-based systems, especially of inferences performed by systems, such as the proposed context-awareness framework, independent of any implementation. The web ontology language (OWL), which is a family of description logic languages (DL) defined by the *World Wide Web* (W3C) consortium for semantic web, is widely adopted in context-aware computing for developing ontology-based context models to infer higher-level contextual knowledge from complex context data. The description language used in OWL to formally model contextual information and their relationship is a category of knowledge representation language for the definition of knowledge bases and for the execution of automatic reasoning over them [96]. Using classes and individuals (instantiations of classes), as well as properties that relate the former in the domain of interest, these elements, together with operations provided by OWL, enable to obtain complex description of the domain of knowledge. In order to develop a formal ontology model, basic DL semantics concepts must be clearly understood. The semantics are given in terms of interpretations τ, which consists of a non-empty set Δ^{τ}, representing the domain of interpretation, and the interpretation function, which assigns to every atomic concept A, a set $A^{\tau} \times \Delta^{\tau}$, and to every atomic role R, a binary relation: $R^{\tau} \subseteq x \ \Delta^{\tau}$.

Typically, the functional description of a reasoning system is specified by pair <T, A>, which represents a TBox and an ABox. The TBox contains intentional knowledge in the form of a terminology of taxonomy developed through declarations that describe the general properties of some concepts. The TBox is composed by a set of axioms in the form of $C \subseteq D$ or $P \subseteq R$ (inclusion) and $C \equiv D$ or $P \equiv R$ (equivalence), where C and D are classes and P and R are object roles or properties. For example, if we defined two classes, namely *UserActivity and WalkingActivity* in the *TBox,* the axiom *WalkingActivity⊆UserActivity* would denote that *WalkingActivity* is a specialization of *UserActivity,* i.e. each instance of *WalkingActivity* is also an instance of *UserActivity.* An axiom $C \subseteq D$ is satisfied by an interpretation τ when $C^{\tau} \subseteq D^{\tau}$.

The ABox on the other hand contains assertions in the knowledge base, i.e. knowledge that is specific to the individuals and their properties in the domain of discourse. ABox contains a set of axioms of the form X: C and <x,y>:R, where x and y are individuals, C is their class, and R is their properties or relations/role. For example, *"Sitting:UserActivity",* which denotes that *Sitting* belongs to the class *UserActivity: <Ana, Sitting>:hasCurrentActivity* denotes that Ana is currently *sitting. Axioms x: c and <x, y>: p* are satisfied by an interpretation τ *when* $x^{\tau} \epsilon C^{\tau}$ *and* $<x^{\tau}, y^{\tau}> \epsilon P^{\tau}$ respectively. An interpretation τ satisfies an *ABox* when τ satisfies all the axioms A, and interpretation τ that satisfies *TBox* and *ABox* is called a model of *<T, A>*. To perform reasoning tasks on domain knowledge *T,* following properties have to be satisfied [96]*:*

(1) Subsumption: A class C is subsumed by a class D with respect to T if and only if $C^{\tau} \subseteq D^{\tau}$ for every model τ of the T.

(2) Satisfiability: A class C is satisfiable with respect to T if and only if there exists a model τ of the T such that C^{τ} is non-empty.

(3) Equivalence: A class C is equivalent to another class D with respect to model τ of T provided that $C^{\tau} \equiv D^{\tau}$ for every model τ of T.

(4) Disjointness: Classes C and D are disjointed with respect to a model τ of T provided $C^{\tau} \cap D^{\tau} = \theta$ for every model τ of T

(5) Consistency: a ABox is consistent with respect to a TBox provided there exists an interpretation of the model <T, A>

(6) Classification: computes the hierarchy of the atomic classes in the TBox.

These basic definitions allow the understanding of basic ontological logic language used by OWL, upon which proposed context-awareness framework is designed.

3.5.2.2 The context ontology model

Having presented the basic principles of ontology, we propose in this section the adoption of the ontology model in the proposed framework for inference of higher contextual information from atomic context classified by the context recognition model. The conceptual context framework incorporates seven generic interrelated concepts, which were

defined for providing context information for context-aware media recommendations domain. Five of the core concepts are based on MULTICAO [153], and extending its core concepts. Whereas two additional concepts have been incorporated into that ontology: Activity and Time. The time ontology was extended from publicly available ontology [146] [143] [178]. MULTICAO was developed using OWL DL with adequate provision for adaptation decisions, relying on various rules based mechanism, which supports an automatic knowledge inference from contextual information. More importantly, MULTICAO was modeled using MPEG-21 UEDs and MPEG-7 as its basic concepts: It was developed as a two layered ontology model.

The first layer is the primary domain ontology with five core concepts: *User, Terminal, Environment, Media and Network*. Figure 3.14 illustrates the relationships between MPEG-21 UED context representation and ontology and SWRL based context model. The ontology model of the proposed framework is based on *MULTICAO*, presenting a two-layer view or design, but with two additional core concepts- user activity and time concepts. Figure 3.15 shows the two-layer model of our ontology.

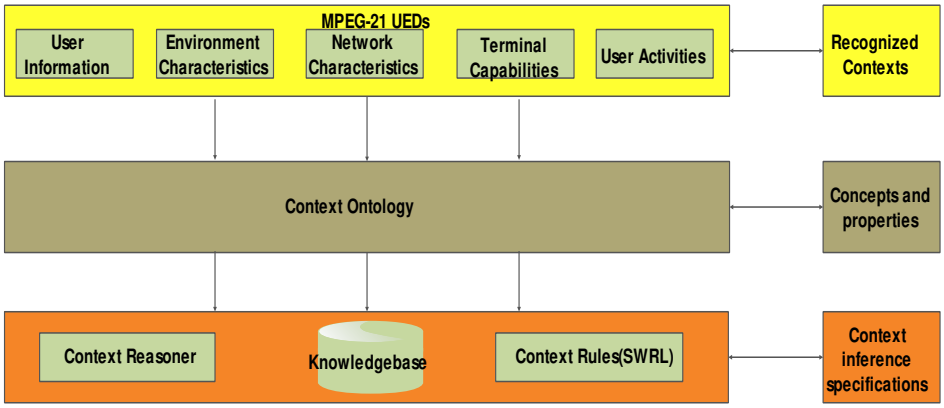


Figure 3.14- Context inference model

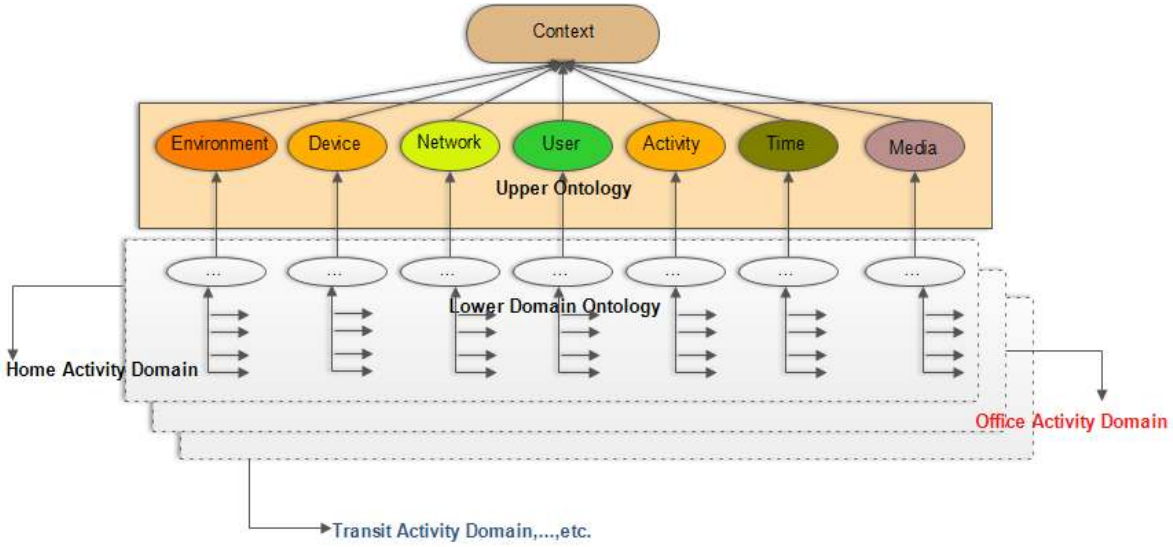


Figure 3.15 - Context ontology model

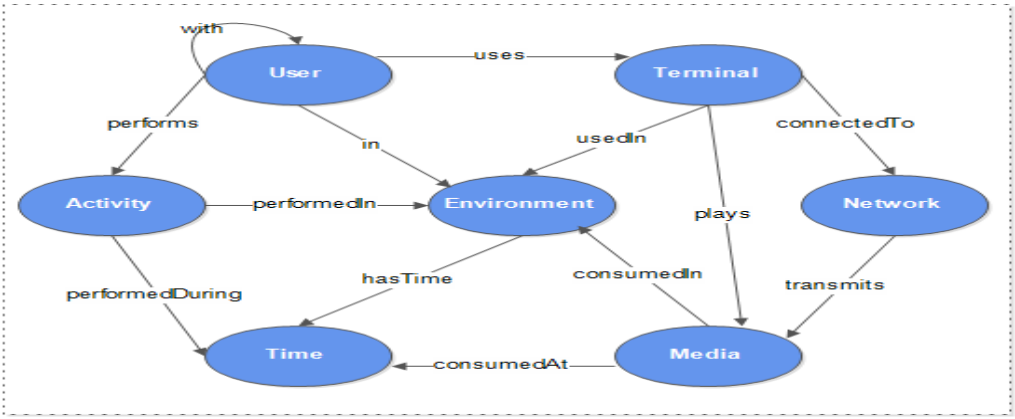


Figure 3.16 -Ontology model

The first layer consists of the basic and generic seven concepts common to all domains of interests generally known as upper ontology. The concepts in this domain are valid for many application scenarios. The second layer incorporates specific domain ontologies, integrating reasoning and rules for inferring contexts specific to each domain. The specific domain is designed based on the combination of user specific activities and locations. For example, user home domain activity related contextual information determines contextual information related to user's activities when at home. Figure 3.16 illustrates the relationships between ontology model's main concepts. In the next section, each of these concepts is briefly introduced.

(1) User concept

The central focus of any context-aware recommendation system is the user. In these systems, information such as location, activities and preferences are very crucial to building a system adapts to user's needs. This contextual information influences user's preferences for media consumption, and therefore, when recommending media items to the user, it is important this information is incorporated into the recommendation process. Contextual information is the most important input when using contexts in the recommendation systems. Contextual information such as user activities, context history, user demographic information such as sex, age, name, etc., as well as user presentation preferences from the elements of the user concept in the ontology model. Figure 3.17 illustrates the user concept class and its associated object properties, and its relationship with other ontology concepts such as Activity, Environment, etc.

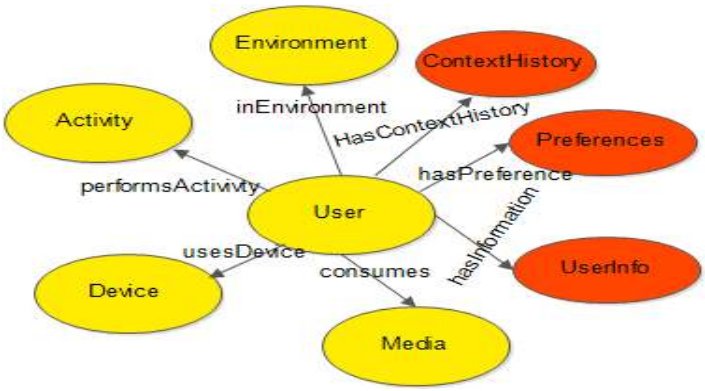


Figure 3.17 - User concept

(2) Device concept

The characteristics and capabilities of user devices can have significant impact on the recommendation process. The device characteristics can be used to adapt recommendations to the characteristics of the user's device. For example, the device battery level can be used to determine the appropriate format in which to present recommended content to the user. In fact, it may present the content in a form that requires less power such as in text form. If the battery level is low, the recommendation system could decrease the brightness or lower the spatio-temporal resolution of the content in case of an audiovisual content. Additionally, this concept also describes the capabilities of the user device.
Information such as device display size, as well as coding format supported by the device is important contextual information supported by this ontology concept. As illustrated in Figure 3.18, this model shows how the device concept relates to other concepts such as user, activity, environment, network etc. For example, to recommend audiovisual content to users, if the size of the user's device screen is smaller than the dimensions of the audiovisual content being recommended, the content can be scaled down to a lower resolution to accommodate its presentation to the user. Figure 3.18 shows the device concept, its associated object properties and its relationship with other concepts of the ontology model.

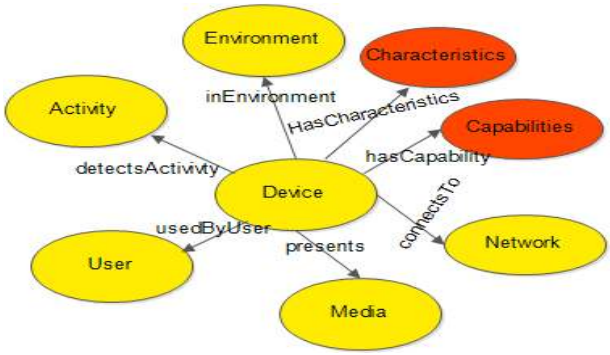


Figure 3.18 - Device concept

(3) Environment concept

Another important concept of the ontology is the natural environment. This concept influences the kind of items users would prefer to consume depending on the environmental condition or situational contexts. It provides information about the environment in which a user interacts with the recommendation system or his device. There are two groups of fundamental concepts related to the environment concepts: location, including logical (e.g., city, street, etc.) and physical (e.g. GPS coordinates) and environment conditions (e.g. noise level, illumination and weather information, etc.) For example, information about the current weather can be used to provide more relevant media recommendations to users i.e. the category and genre of contents preferred by the users can be influenced by weather information. Alternatively, if for example, the noise level is too high in a given environment, the system can automatically increase the volume of recommended music being currently played by the user. Figure 3.19 presents the basic design of the environment concept, its associated object properties and its relationship with other concepts in the ontology.

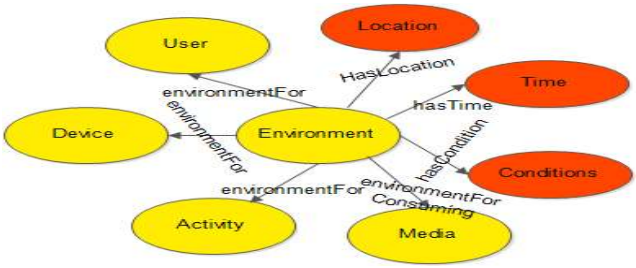


Figure 3.19 - Environment concept

(4) Media concept

Information about candidate media content to recommend is also an important concept. This concept has been fully defined in the *MULTICAO* ontology [146], [153] based on MPEG-7 MDS descriptions. The most important features of the class are media coding, audio coding, visual coding as well as the presentation format. From this concept, information about the genre, titles, etc. of the content can be obtained. This information could be used in the recommendation process to filter important characteristics of the audiovisual content, such as video audio formats, etc. However, in practice, especially on the web, most content is not described or represented using MPEG-7, therefore, we proposed to use the description APIs such as Web services to obtain descriptive /metadata information about candidate content. Figure 3.20 shows the media concept, its associated object properties and its relationship with some of the concepts of the ontology.

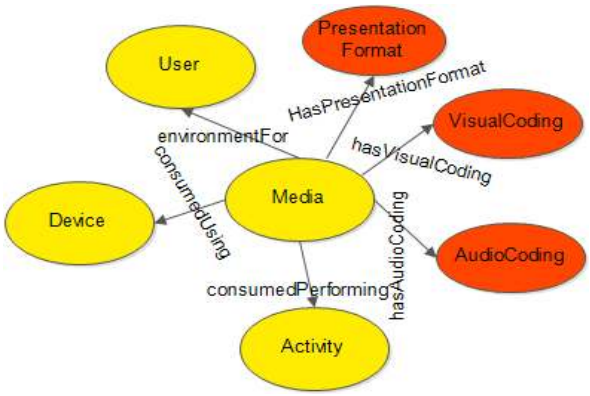


Figure 3.20 - Media concept

(5) Network concept

The network concept describes the conditions and characteristics of the network connection of the user device. It describes such aspects as the network maximum bandwidth, network minimum guaranteed bandwidth, error rate, network delay formation and currently available network bandwidth.

These characteristics can be used either to enhance a minimum level of quality of media consumption or even to help to decide between different versions of the same media item.

For instance, when recommendation such as movie is being played by the user in full quality mode and suddenly the network bandwidth drops to a level that would not support the continuous playing of the movie, then a lower resolution version of that movie can be delivered instead of stopping the movie from playing due to inadequate bandwidth. Figure 3.21 illustrates the network concept and its relationship with the user's mobile device.

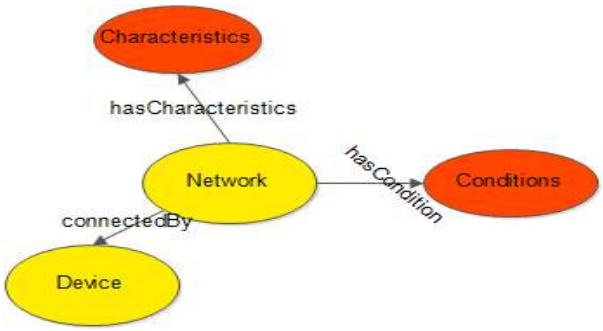


Figure 3.21 - Network concept

(6) Activity concept

User activities can be used to in addition to other context information in recommendations given that they have a strong impact on the preferences and needs of mobile users, as previously stated. As explained in the previous sections low-level to address this problem, low-level data obtained from device-embedded sensors is channeled to a context recognition model running on the device, which recognizes high-level activity being performed by the user. In order to relate this activity information with other contextual information, an activity concept was designed as part of the ontology. These activities are sub-classed from three main concepts: Still activities, which are those activities that do not involve a user's change of positions or location, e.g. sitting, standing, or lying. Motion based activities are those that involve changing of positions and locations of users, e.g. jogging, walking, running, etc. The third subclass are those activities in which the user is practically still but his/her position moves, such as when a user is sitting a moving vehicle.

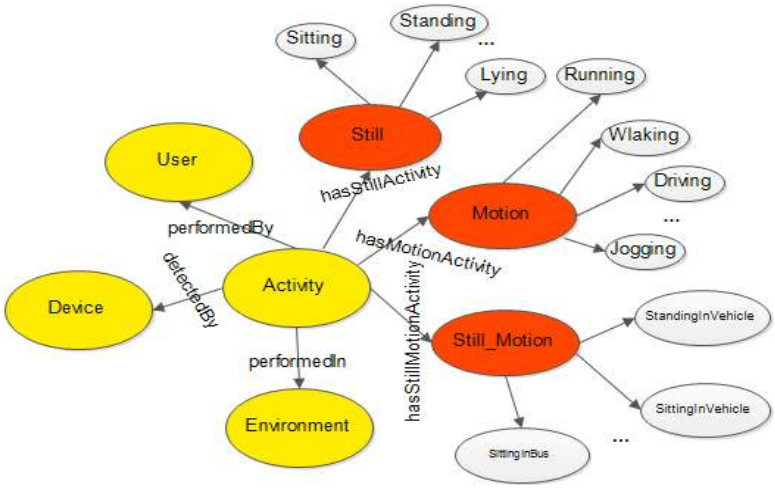


Figure 3.22- Activity concept

Examples are sitting or standing in a vehicle and driving. Based on logical location (which is one the concepts that relate the user activity to the environment where such activity is performed), six classes of activities were defined, namely, *Home, Study, Office, Sporting, Leisure,* and *Transit*. These six classes can belong to one or more of the three main subclasses of the main activity class. Sporting activities occur when the user is not at home and performing some kind of sport related activity, either indoors or outdoors. The transit-based activities include traveling in a bus, train or car. The leisure-based activities involve things like sightseeing or going to a concert. Figure 3.22 shows the activity concept and its relationship with user, device, and environment concepts.

(7) Time concept

The time concept as shown in Figure 3.23 is primarily used to capture the "when question" of the other concepts. Time duration, start time, end time, etc., are some of the elements of this concept. These elements of time concept are used to infer, for instance, high-level time information such as weekdays or weekends that are subsequently used in addition to location information to infer a user's contextual situation including his activity contexts.

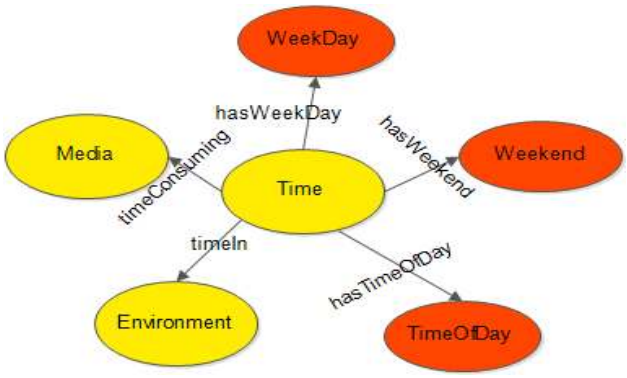


Figure 3.23- Time concept

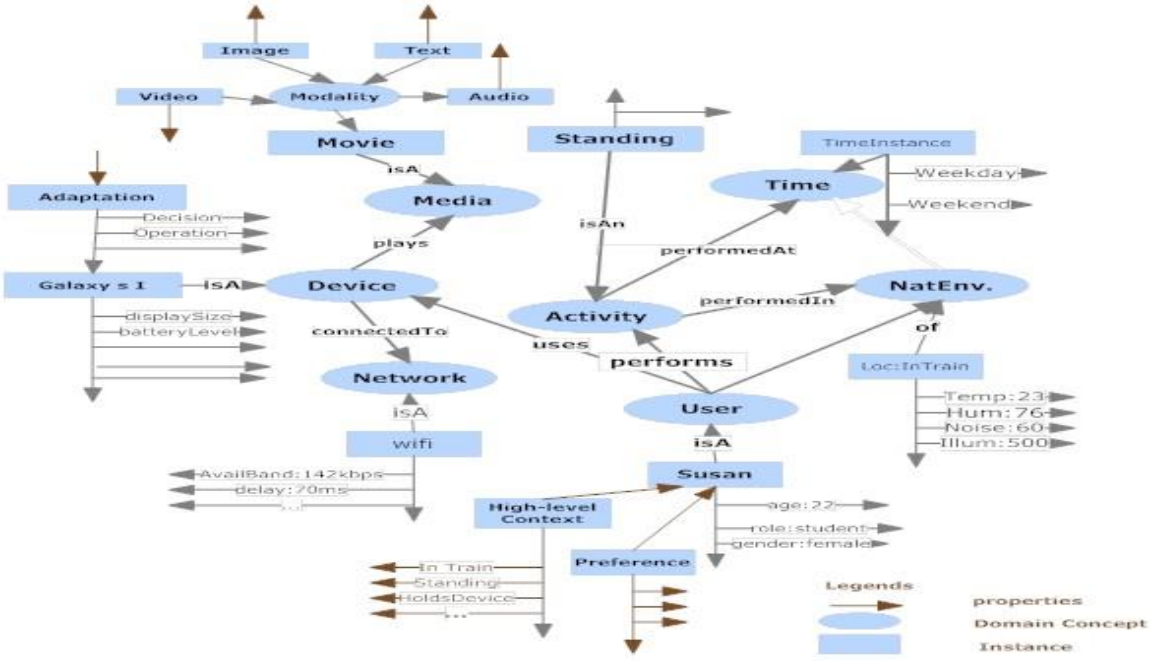


Figure 3.24 - Example ontology instantiation

3.5.2.3 Context reasoning using SWRL

The knowledge-based context model incorporates reasoning mechanisms based on two methods. The first method is the inherent ontology reasoning mechanism that is responsible for checking class consistency and implied relationships. This method uses the inference engine Pellet [175] for inference purposes. The second method is based on the Semantic Web Rule Language (SWRL) [129], [165]. Because OWL does not provide mechanisms for expressing all relations between concepts in ontology model, OWL was extended with SWRL to allow inferring new knowledge from multiple facts or conditions at the same time by providing mechanisms for expressing complex relations [207]. For example, OWL cannot express the relation between a child and married parents because it does not have the mechanism to express the relation between individuals with which an individual has relations. For instance, OWL cannot relate that a user, say Ana, is the child of married parents James and Comfort. Nevertheless,

with SWRL, this can be expressed in the following way: $user(?u)\hat{}hasParent(?u,?f)\hat{}hasSpouse(?f,?m) \rightarrow childOfMarriedParents(?u)$.This specification adds rules to OWL ontologies, while also providing an extra layer of expressivity and the capability to infer additional information from an OWL knowledge base [218]. Basically, SWRL consists of antecedent and consequent parts, which in turn consist of positive connections of atoms. Informally, SWRL can provide high level reasoning in the form of *if all atoms in the antecedent are true, then all the consequents of the rule must be true or vice versa*. Additionally, SWRL comes with built-in unary predicates for describing relations between classes and data types, binary predicates for data properties and some n-ary predicates such as arithmetic operators to compute desired behaviors. For example, it is easy using SWRL to infer if Ana, in the previous example, is an adult or not using these built-in predicates thus, $user(?u)\hat{}hasAge(?u,?age)\hat{}swrl:greaterThan(?age,18) \rightarrow adult(?u)$ likewise by combining time and location data, together with information on the type of device being used and whether the user is accompanied or not, as well as user activity, the system may be able to infer the situational activity of the user e.g. *Ana is sitting at home.*

Listing 1
multicontext:hasUserId(?user, ?userId) ∧ multicontext:atHome(?user, ?location) ∧
multicontext:weekEndAtLocation(?location, ?weekend) ∧ multicontext:locationTimeIs(?location, ?time) ∧
multicontext:isMorning(?location, ?time) ∧multicontext:inIndoorLocation(?user, "true") ∧
multicontext:hasUserActivity(?derived, ?activity) → multicontext:hasHomeLocationActivity(?user, ?sitting)

Listing 2
multicontext:hasUserId(?user, ?userId) ∧ multicontext:atOffice(?user, ?location) ∧
multicontext:weekDayAtLocatiom(?location, ?weekday) ∧ multicontext:locationTimeIs(?location, ?time) ∧
multicontext:isEvening(?location, ?time) ∧ multicontext:inOutdoorLocation(?user, "true") ∧
multicontext:hasUserActivity(?derived, ?activity) → multicontext:hasOfficeLocationActivity(?user, ?walking)

Listing 3
multicontext:hasUserId(?user, ?userId) ∧ multicontext:atOffice(?user, ?location) ∧
multicontext:weekDayAtLocatiom(?location, ?weekday) ∧ multicontext:locationTimeIs(?location, ?time) ∧
multicontext:isEvening(?location, ?time) ∧ multicontext:inOutdoorLocation(?user, "true") ∧
multicontext:hasUserActivity(?derived, ?activity) → multicontext:hasOfficeLocationActivity(?user, ?sitting)

Figure 3.25 - Listings 1, 2 and 3: Example rules for inferring higher-level contextual information

Figure 3.24 provides an example of specific instantiation of an individual user's ontology model. In listings 1-3 of Figure 3.25, we provide examples of specific domain rules supported by the context ontology model. Listing 1 is an example that infers that the user is indoor, sitting at home, on a weekend. In listing 2, the user is in outdoor in the office walking, whereas, in listing 3, the user is indoor, sitting in the office. Using this contextual information, personalization systems can designed to recommend relevant media items that suit these specific situations of the user.

3.6 Summary

This chapter has described a conceptual framework developed to provide context awareness functionality for context-aware applications notably the context-aware personalized recommendation system proposed in this thesis. This framework consists of a context recognition model, an MPEG-21 UEDs and MPEG-7 MDS open standard context representation and an OWL ontology based inference model with SWRL rules for inferring high-level context information from the user's device. The context framework's design supports dynamic acquisition, representation, inference, reasoning, storing, sharing and consumption of contextual information. It provides processes that can be used by context-aware applications, with platform neutral functionalities. The framework addresses aspects of interoperability by adopting standardized data schemes and semantic techniques thus enlarging the scope of its application. Additionally, its functionality is provided in a seamless and non-intrusive way, as the context-awareness framework automatically obtains low-level context data from the mobile device's built-in sensors. These data are the preprocessed to minimize noise, and to extract features for discriminating between various contexts using sliding windows and statistical features. The features are provided as inputs to the classification algorithm that infers the context represented by the features. The recognized context information is represented using MPEG-21-UEDs, which

are then instantiated into the ontology together with other low-level context data to infer high-level situational contexts of the user. This high-level contextual information can be consumed as web services by any device with any operating platform. In chapter six, the use of this conceptual framework by a context-aware media recommendation application is demonstrated. Additionally, in chapter six, evaluations of the developed context framework, especially its context recognition processes are presented.

Chapter 4

Conceptual Framework for Context-Aware Multimedia Personalization

4.1 Introduction

As discussed in chapter two, existing traditional recommendation systems do not fulfill the requirements for true and proactive context-aware personalized recommendations. Many of these systems lack the capability to adapt recommendations to user's dynamic contexts, and when they have, such capability is limited, static and often intrusive. Based on the principles of context-awareness developed in this thesis and described in chapter three, we have designed a conceptual framework for context-aware multimedia recommendations to mobile users, which we call Conceptual framework for Context *A*ware Personalized Mobile Multimedia Recommendations (CAMR). This chapter describes the principles and conceptual architecture of such framework, identifying requirements for personalized mobile multimedia recommendations.

The critical challenge of traditional personalized recommendation systems is the lack of capability to obtain dynamic contexts to filter user preferences during the recommendation processes, since they rely on ratings provided by users [29]. However, in mobile environments it is difficult to guarantee the availability of such ratings, especially in all diversity of contexts that may occur. Apart from relying on users to provide explicit contextual information, most existing context-aware recommendation systems wait for users to initiate the recommendation process, and thus they can be regarded as static or explicit recommendation systems [179]. These problems and considerations have motivated the development of CAMR. CAMR relies on the context awareness framework developed and fully described in chapter three, which identifies the contextual information dynamically with little or no user intervention. CAMR incorporates contextual information into traditional recommendation processes using a contextual user profile model. Additionally, to address the cold start problem, CAMR uses contextual information of like-minded users to provide recommendations. The hypothesis upon which such solution is developed is that users with similar contexts usually share similar interests. Thus, a user who has never been in a specific context or who is using the system for the first time can still receive recommendations based on what other users in similar contexts have preferred.

In this chapter, we focus our discussion on a generic conceptual context-aware framework for personalized media recommendations as illustrated in Figure 4.1. This framework defines a set operations and algorithms that support context-aware recommendation systems, especially in mobile environments.

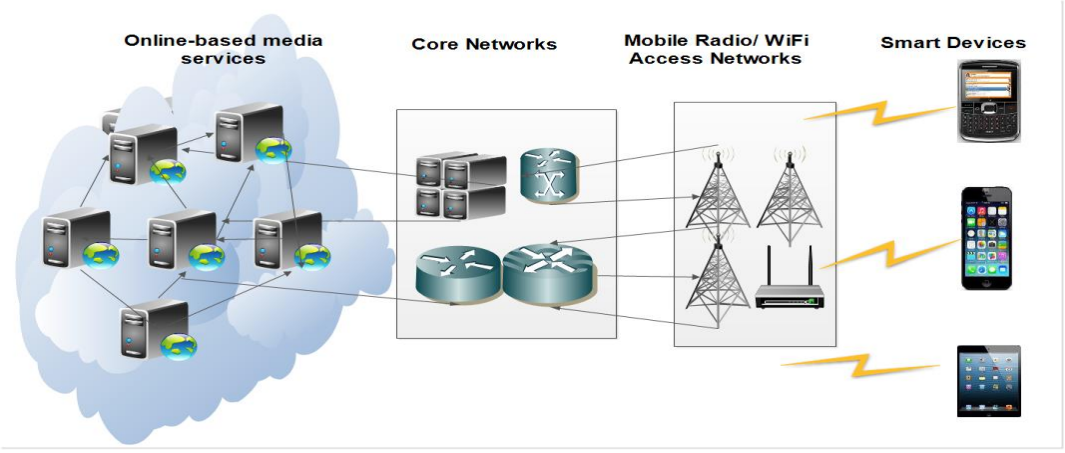


Figure 4.1- Multimedia consumption via smart devices in mobile environments

4.2 CAMR requirements

Providing media items to users in a heterogeneous mobile environment remains a difficult challenge [5]. In fact, enabling proactive contextual suggestions and delivery of relevant media items that satisfy user's preferences, in such environments, require careful considerations. This entails addressing a number of key issues, among which are dynamic user context recognition and representation, contextual user profile and preference management, implicit recommendation of relevant multimedia items and adaptation of media item presentations to device constraints and network conditions. Besides, developing a framework that incorporates these aspects in a unifying way has never been a trivial issue. In this section, we introduce the requirements for the development of the proposed conceptual context-aware personalized recommendation framework.

4.2.1 Context recognition and representation

A multimedia item is considered relevant if it suits the preferences of the user in the context of the user's active interaction [36]. Therefore, contextual information is an important factor that can help to determine the relevancy of multimedia items in mobile environments. If the context is wrong, this may lead to wrong or irrelevant recommendations. Additionally, contextual information can be used to address the cold-start problem. The coldstart problem is a product of inadequate rating information owing to either new user problem or inability of users to provide rating information when they consume content [29]. To exploit contextual information for personalized content recommendations, it is important to identify accurately the user's contextual situations, requiring dynamic context sensing and recognition, using mobile devices. Existing systems have always used an additional external hardware sensor to provide contextual information or in most cases burden users with the responsibility to statically provide their own contextual information. A context awareness framework meeting these requirements, including the details of how user's device can be used for the dynamic context recognition and representation has been described in chapter three of the thesis.

4.2.2 User profiling

Another key requirement to offer context-aware personalized recommendations to users in mobile environments is the user personal profile. User profiles should encode all desirable media content features, customized for the profile owners taking into consideration the user's contexts. It is important to note that traditional user profiling techniques do not consider context information in the profiling process. The user profile includes information: 1) preferences of the user for desired media content, which can be represented by semantic metadata or other feature of that preferred content; 2) *optional user identities such as names, gender and profession*; 3*) preferred location information*; 4) *usage history*, (5) *user's device information*; 6*) network profile information*; and 7) *other high-level contextual information* such as user activities in relation to user's preferences. Dynamic acquisition of user preference information, user consumption history (if any) and their incorporation with user's contexts to update the user preferences has remained a challenge. Contextual user profile process, which is one of the main components of CAMR, will be discussed in section 4.4.

4.2.3 Media classifications and recommendations

As stated earlier, the sheer size of online content considerably hinders the process of finding useful information, revealing to be a greater handicap for mobile users. Additionally, the notion of "*usefulness*" or "*relevancy*" is very likely to change according to the user's contextual situation, which varies in mobile environments. Accordingly, the ability to correctly identify and classify content (including that consumed by the user) as well as to correctly identify contextual conditions, and associate the two types of data assumes great importance. Media classification requires the extraction of relevant features from the content or otherwise must rely on existing semantic metadata associated to the available content. A system that uses such media characterization to select relevant content to recommend to users needs to identify the characteristics of the metadata upon which it is going to make the selection.

In CAMR, we have defined a data model for representing these content characteristics, designated multimedia content profiles (described in section 4.4.2). The identification and classification of contextual situations has already been described in chapter three. Finally, the requirements imposed on the recommendation process itself is that it should use suitable contents given the contextual situation of the user. The media classification and all the contextual recommendation processes are discussed in sections 4.4 and 4.5.

4.2.4 Content adaptation

The dynamic nature of mobile devices provides situations whereby device and network resources are unlikely to match the required resources for satisfactory delivery of recommended content. One of such situations is when available wireless network bandwidth is not adequate for delivering rich content such as video to the device or when the device does not support the format of the recommended content and thus cannot play the content. In this case, the video's spatial and temporal resolutions can be scaled down to match the available bandwidth or presenting the content in modality such as text or image that does not require much bandwidth for delivery can be presented [6]-[7], [42]. Although this aspect of the project is not the primary focus of the thesis, in section 4.6 the thesis provide some insight on how the work developed in the thesis can benefit from the elegant solution that was already developed in [153].

4.3 The goal and objectives of the conceptual framework

Having provided the basic requirements necessitating the development of the proposed conceptual framework, its main goal therefore, is to contribute solution meeting those requirements. The goal is to provide a generic conceptual context-aware recommendation framework capable of providing relevant recommendations, with minimal user interventions, based on user's dynamic contextual situations and preferences.

4.3.1 Objectives

Considering the limitations of traditional and existing context-aware recommendation systems, and to support and achieve the goal stated above, and having developed a context-awareness framework in chapter 3 for providing dynamic contextual information in the CAMR framework, the following guiding objectives have been identified and expressed in the development of the proposed conceptual framework.

The first objective is to define and develop a dynamic contextual user profile model with capability to relate user's contextual information and preferences to infer unknown preference in the present contextual situation.

The second objective is to the contextual user profile to extend the traditional recommendation processes with capabilities to generate recommendations according to users' current contextual situations with limited intervention and without relying on explicitly provided ratings and contexts.

Third objective is the proposition of an optional adaptation of recommendations to the constraints imposed by the user's device characteristics and network conditions.

4.3.2 Basic framework assumptions

In the development of the proposed framework to realize the objectives state above, some basic assumptions have been considered, which are discussed below:

First, we assume that the proposed framework can serve different kinds of heterogeneous devices, including, mobile phones, tablets, laptops, desktops, etc., depending on which device is available to the mobile user at any given time and location. Therefore, the definition of devices includes these terminals.

Second, the framework operates on the assumption that MPEG-21 based adaptation decision taking engine (ADTE) and adaptation engine that can tailor the recommended content to the user's device constraints and network conditions are readily available [6]-[7].

Third, the proposed framework assumes that Web-based multimedia content's descriptions are based on the MPEG-7 descriptors [48]. From experience, Web based multimedia content metadata is usually not in a MPEG-7 MDS format. Nevertheless, to address this limitation, the framework provides an interface to interact with publicly available APIs for obtaining media item metadata (discussed in chapter 5).

Fourth, the framework additionally assumes efficient and secure interactions of users with the system, which guarantee the trustworthiness of the framework. The framework should however ensure that users have the freedom to choose the information to share by providing preference setting on the user terminals where such preferences can be pre-determined. It adopts anonymity and pseudonimity of users [180], so that the framework has limited knowledge of who these users are in terms of names, sex, etc. at the same time accommodating those users who do not mind to disclose such information.

4.3.3 Framework development principles

In view of the considerations of the goal and objectives of the framework, its development is guided by four design principles. These principles are introduced in the following section. Note that design principles *a* and *b* are addressed in this chapter, whereas *c* is addressed in detail in chapter 5.

(a) Personalized recommendations are governed by active user's current or past (if any) contextual situations and those of other users who are similar to the active user. Existing context-aware recommendation systems have a limitation of relying on users to provide their contextual information explicitly, in addition to relying on user provided ratings before recommendations could be generated. Thus, the new user problem continues to be an issue even in context-aware recommendation systems (CARS), because users with no or inadequate ratings cannot be provided with relevant recommendations. To address this limitation, the basic operation of the proposed CAMR framework is guided by implicit recognition of user's contextual situations, based on a context awareness model described in chapter 3.

(b) Determination of user's preferences based on the implicitly recognized contextual information without asking users at runtime to provide the preference information. This is a very critical part of the framework. The ability of the system to relate user's contextual information, especially their current contextual information with preferences to obtain user preferences is an important consideration.

(c) Allowing some level of control to the user is also an important criterion. It is important to note that while designing context aware recommendation systems with limited user interventions, the designers must provide a balance that allows users to have a sense of control over the system. Consequently, the system should not always, for example, collect user's sensitive information, especially the information that may expose users' to privacy and security risks. Thus, it should afford the user the freedom to either allow or disallow implicit collection of such information by providing users with mechanism to override the complete implicit acquisition of user data in cases where users are not comfortable with a system having access to their private data such as location information. In this case, provision should be made to let users know about the acquired contextual information.

4.3.4 CAMR high-level architecture

The high-level architecture of the proposed framework as depicted in Figure 4.2 highlights the most important components for context-aware media recommendations to meet its design requirements. Figure 4.3 illustrates the cyclic operations of the components, from context sensing and recognition to context-aware recommendations and adaptation. In the architecture, we show how it integrates and addresses the issues discussed in section 4.2. The context management consists of context awareness framework, which detects and recognizes basic user contexts, and a context inference manager, which derives additional contextual information by establishing relationships between different basic contexts identified by the context and activity recognizer as described in chapter three. The contextual user profile manager handles the management of the contextual user preferences and the user interactions with the framework in conjunction with the user feedback manager. The feedback management is responsible for tracking user's interaction as well as updating user's preferences according to his contextual consumption. It also allows users some level of control over the system as it provides an interface for user to specify some basic preferences, as well as allowing or not the system to use some information, e.g. names, sex, age, etc. Additionally, it addresses two challenges. First, it obtains from personal profiles other users' contextual information similar to the active user's current contextual situation. Second, it uses this information to infer the users' preference.

The media profiling management is responsible for obtaining media item metadata from the Web, and processing it into a form that can be utilized by the recommendation management. The recommendation management is responsible for processing, generation, and ranking of context-aware personalized recommendations.

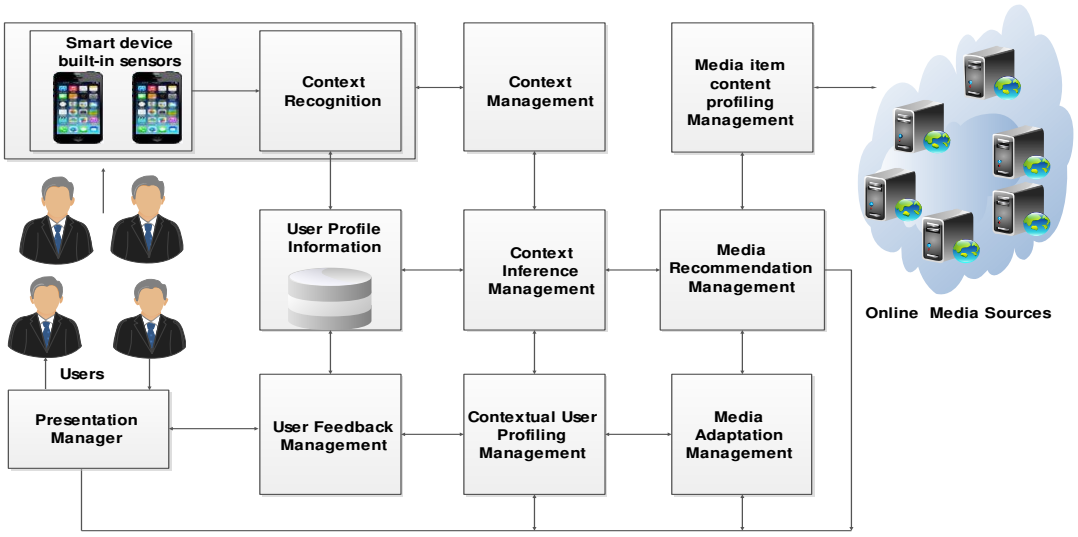


Figure 4.2- CAMR conceptual frameworks showing some of its basic functional parts

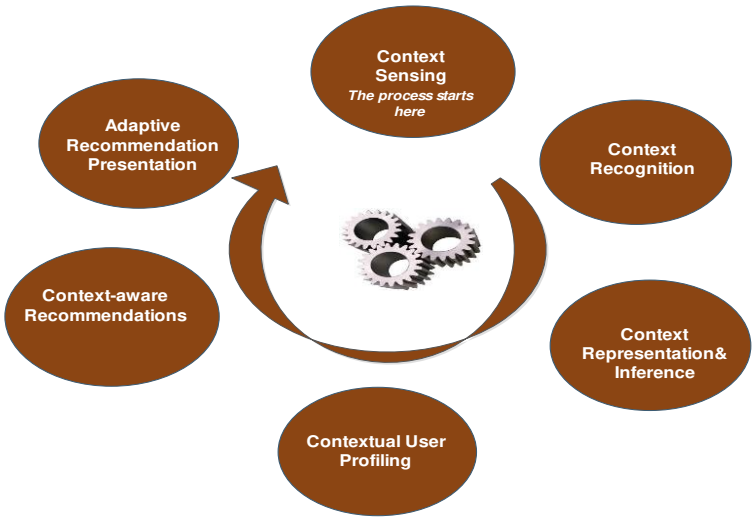


Figure 4.3-Proposed framework process cycle

4.4 Contextual user profile

Universal Multimedia Access (UMA) researchers aim to allow consumption of any content anywhere and at any time, by adapting content to contextual constraints [139]-[153]. Nevertheless, as discussed earlier in the previous chapters, many of the proposed solutions do not anticipate user's preferences according to her contextual situations. In addition, those that consider contextual preferences take into account only a limited and static set of contextual information [14]. We argue that these solutions are passive concerning the selection of content they deliver to users because they assume that context information is readily available, and sometimes they rely on asking the user to supply manually this context information.

Thus, these systems require explicit user interactions before they can infer user preferences. While multimedia recommendation system's main goal is to help users find relevant multimedia items that match their preferences, they must also aim to limit intervention of users during recommendation processes. However, in a context-aware recommendation system, the efforts required to use the system must be much less than the benefits users reap in return. A truly context-aware recommendation system requires minimal user interventions while performing the recommendation process. To achieve this, a user profile model must be dynamic, and should require minimal explicit inputs from users. Having provided a framework to obtain a user's contextual situation in chapter three, here we provide a solution that uses contextual information obtained from the context-awareness framework to determine user's preferences in such contextual situations.

In this section, the chapter describes an approach that addresses this problem, which anticipates user's needs, based on user's implicit contexts. We describe the profiling process and the context-aware recommendation processes developed as part of the solution for mobile users. The approach adopts case-based reasoning as a methodology, using nearest neighbor model [83]-[84].

In case-based reasoning methodology, solutions to previous problems are used to solve a new problem. In other words, the solutions to past problems are used to solve newly identified problems, utilizing the specific knowledge of previously experienced problem situations called, cases [156]. Case based reasoning or CBR is a machine learning method that solves a new problem by remembering a previous similar situation and by reusing this information and knowledge in new situations [156]. It relies on the general knowledge of the problem domain or making association along generalized relationships between problem description and conclusion. One unique and important feature of CBR based method is its ability for incremental and sustained learning because a new experience is retained each time a problem is solved, with immediate availability for use to solve future problems [83]. Case-based reasoning method provides the methodology to solve a problem, but not the technology to solve it [84]! Thus, the concrete solution or implementation of the provided solution is left open, depending on the problem being solved. As reviewed in chapter two, section 2.2.5.6, CBR has been used extensively to solve personalization and recommendation problems. In a context-aware media recommendation system, CBR can be used in the contextual user profiling to infer user's contextual preferences in the user profile knowledge base, based on the currently recognized user's contextual situation [83].

To accomplish this objective, CAMR framework adopts four main processes corresponding to the CBR cycles. These cyclic methods are: retrieval of contextual preferences, reusing of contextual preferences, revising contextual preferences, and updating contextual preferences. Our approach is different from how CBR has been employed in recommendation systems. For example, the system presented in [156] relies on experiences of the active user to predict his present preference. This solution still does not provide how to obtain preferences of a user who does not have any previous experience using the system. In this thesis, we address this problem by using experiences of other users whose contexts are similar to the context of the new user to obtain his/her contextual preferences. In the next subsections, we describe how to identify users' contextual preferences using contextual information based on case-based reasoning methodology to minimize user's interactions with the system.

4.4.1 User profile definition

CAMR defines and identifies user preferences using user's contextual information. Because user's preferences differ in different contextual situations, it is important that the user preferences are identified and their relationship with the user's contextual situations clearly defined. In the recommendation processes therefore, the selection of relevant user preferences according to the user's current contextual situation follows the specification of the relationship between the user's preferences and contextual situations. To utilize contextual information to obtain user's present preferences, the following processes as illustrated in Figure 4.4, are defined and captured in the contextual user profiling process. Roughly mapped into the four cycles of CBR based method described in chapter two, these processes are:

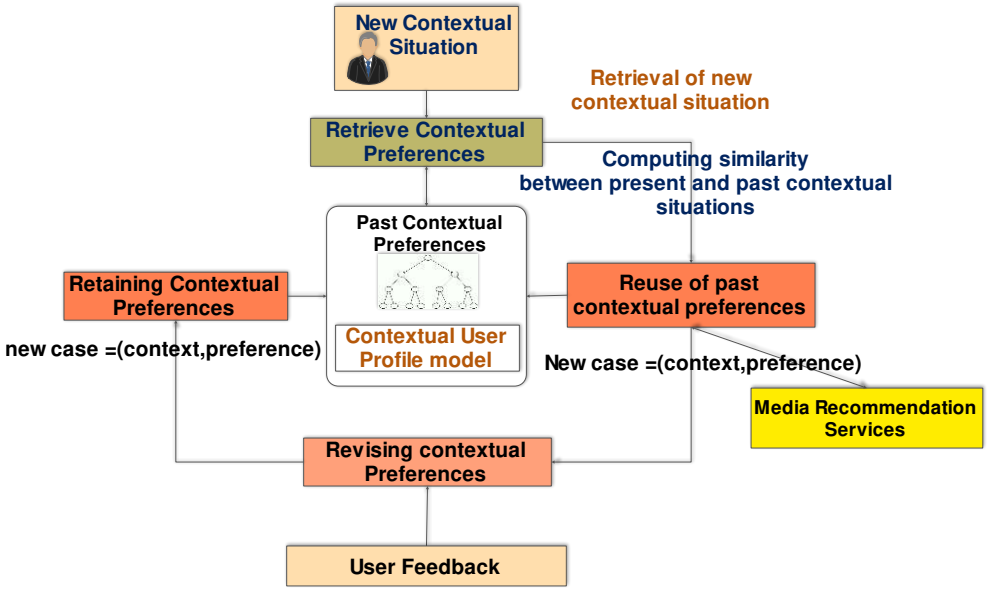


Figure 4.4- Contextual user profiling model

(a) *Identifying user's present contextual situation*. Since user preferences are not fixed, and are characterized by contextual situations, these contextual situations are identified and obtained automatically from user's environments. (b) *Retrieving most similar contextual situations either from the user's context history (if any) or from those of other users*. This is an important phase of the contextual user profiling process. To determine the expected preferences of the user in the current contextual situation, it requires a method or algorithm for determining the degree of similarity between contextual situations. There are many possible methods that can be used e.g. nearest neighbor algorithm or other machine learning algorithms. This thesis adopts nearest neighbor algorithm because of its speed, simplicity and accuracy [74]. (c) *Reusing the contextual information obtained from* (b) by extracting the preferences in those identified contextual situations to determine the active user's preferences in the current context. After similar contextual information has been identified and retrieved, the system then obtains users' preferences in those contextual situations and uses them to identify the current user preference in the current contextual situation. This phase of the profiling process is actually carried out in the recommendation process, which is discussed in section 4.5. It works on the assumption that users express similar preferences for multimedia content in similar contextual situations. (d*) Revising contextual preferences used in* (c). It is important to point out that user profiling is a continuous process, even after users have been provided with recommendations. Whether the recommendation provided is accepted or not by the users, the system keeps track of the user's response, and appropriately updates the profile according to this new information.

Thus, in this phase of the user profiling process, direct/ indirect inputs or feedbacks from users are processed to improve the quality of future recommendations. For example, this phase consists of user feedback, obtained either explicitly or implicitly. In explicit feedback, user's acceptance or rejection of the recommended items is obtained via the selection of recommended item in the system's user interface. In the implicit feedback process, the user is not directly involved; rather, the system monitors the user's interactions with the system when provided with the recommendations and keep track of user's important interactions. For example, it could monitor the number of time user has preferred certain items belonging to a specific category of content, e.g. movies that the user usually prefers such as an action movie or otherwise, and in what context users always prefer such genre of movies.

(e) Retaining the revised contextual preferences. In this phase, after the feedback information is obtained from the user, information, including the contextual information and user preference is stored in the user profile repository for future use. These processes are discussed in the next sections.

4.4.1.1 Contextual user profile representation

A user profile describes user preferences as a summary, normally based on the history of the user's actions [23]. CAMR's contextual user profile model summarizes the user's content consumptions into a limited set of categories, where categories are characterized by one or more genre, and a number of properties characterizes the genre. Additionally, it incorporates contextual dimensions, associating one or more inferred context to each *category-genre-property* concept (which is defined shortly). It represents each user profile as a four level tree-like structure as illustrated in Figure 4.5, with the root of the tree representing user's optional demographic information. The first level nodes correspond to the content category, the second level represents the content genre, and the third level contains the properties of a given category-genre. This level provides the multimedia item's metadata, characterizing at a finer detail, the consumed content and thus, the user preferences.

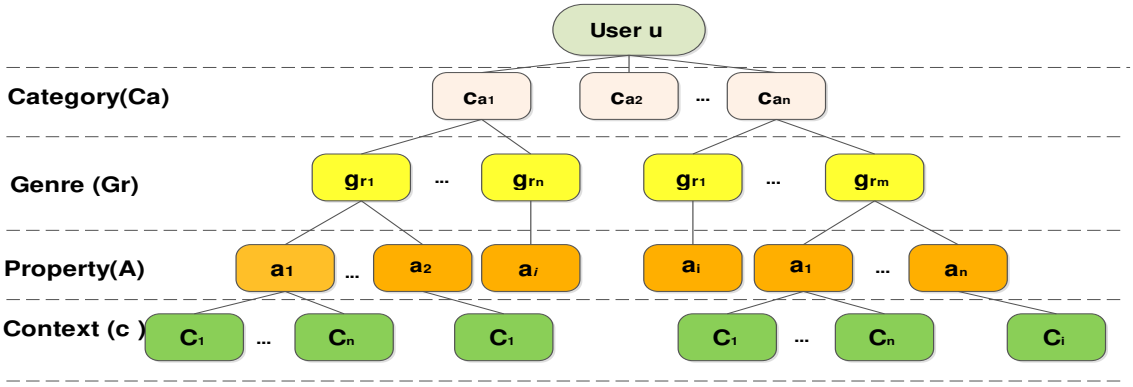


Figure 4.5- The generic contextual user profile representation

A limited set of properties characterizes each genre to obtain a good compromise between sufficient degree of characterization of the content (hence, sufficient ability to make distinctions) and a reasonable dimensions of the user profile. The leaf nodes provide information about the contexts where the user preferences have been observed. These leaf nodes have four fields – *type*, w*eight, intensity,* and *lifetime* – whereas all other nodes have only the type field. In the leaves, the types represent the type of context. The weight represents the preference of the user for a *category-genre-property,* obtained by tracking the number of times users have consumed item that matches a given category-genre-property. The intensity and lifetime track user's contextual consumption history to enhance multimedia personalization. The lifetime is the time (e.g. in Hours, days, moths, etc.) since a target user consumed item of the *category-genre-property*. Using these weighted parameters, the system is able to structure the multimedia content in relations to the contextual preferences. The *intensity* provides information about the number of times the user has consumed items of that *category-genre-property* and in a specific context. This information is used as an implicit feedback to update the user profile in order to improve subsequent recommendations. The intensity (the dynamic preference of the user) is computed by summing up the product [weight x lifetime], given in equation (4.1), of all their occurrences. The procedure for processing and retrieving user's dynamic preference (intensity) is elaborated with an example in section 4.4.2(d), reusing contextual situation to obtain the contextual user profile vector.

$$p_{i,c} = \sum_{i=1}^{n} (weight * lifetime) \qquad (4.1)$$

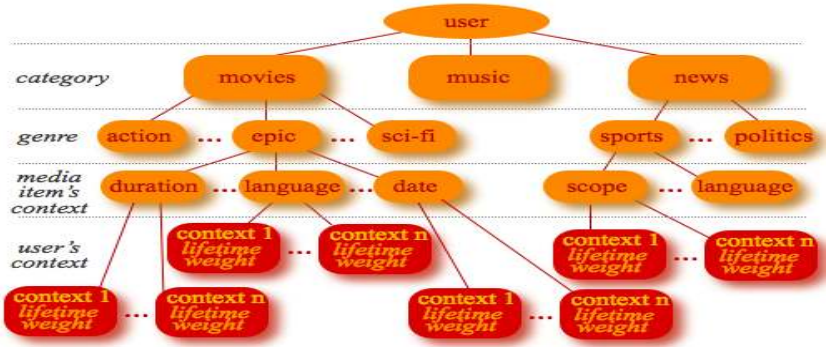


Figure 4.6- An example contextual user profile representation for movie, music and news

The definition of user profile model as introduced in the last section presents a hierarchical structure, which structured as a four-level tree-like structure. Figure 4.6 shows a hypothetical contextual profile model, depicting three categories of media content, namely, movies, music, and news with associated genre, properties and contextual information. The information in the profile is used to build the contextual user profile. The elements of the user profiles are computed in a filtering process, using contextual pre-filtering method [73] and case based reasoning approach using a nearest neighbor algorithm to retrieve and to learn the user's contextual preferences as described in the next sections [83].

4.4.1.2 Contextual user profile formal definition

As depicted in Table 4.1, let $U = \{ u_1,u_2,u_3,..u_n\}$ be a set of N mobile users. Each user u_i is defined by a set of preferences $Pu = \{Pu_1, Pu_2, Pu_3,..., Pu_n\}$, where each preference Pu_i of user u_i is further defined by other sets of attributes. For instance, from the above definition, a user u_i where $u_i \in U$ is characterized by a set of demographic information A and a set of preferences Pr. Each preference is classified into a category, $Ca = \{cr_1,cr_2,cr_3,...,cr_n\}$ and each category is characterized by a set of genres, $G = \{Gr_1, Gr_2, Gr_3,..., Gr_n\}$, and each genre is characterized by a set of attributes properties $A = \{a_1, a_2, a_3,..., a_n\}$. Additionally, let us define a set of high-level contextual information, $C = \{c_1, c_2, c_3,..., c_n\}$ associated with each preference Pu_i, where C_t (t ϵ 1…n) is a context type t. The context type could be a location of a user when he consumes a content. This context type is further characterized by other contextual information such as activity the user performs at that location, the time at the location, the weather information, the illumination etc. It could also be the characteristics of the user's device and networks.

In the above definitions, context C has a very complex structure, reflecting the nature of the difficulties to represent contextual information concisely. However, we assume that contextual information is defined by a structure that allows atomic contexts are represented in a manner that allows higher contextual information to be inferred automatically.

Table 4.1- User profile definition

$PR_{(ui)}$	User's Optional Demographic Information			User Preferences			User Context Type		
$\begin{cases} p_{r1,1} \\ p_{r1,2} \\ p_{r1,r} \end{cases}$	d_1	…	D_m	Pu_1	…	Pu_n	c_1	…	c_n
	d_1	…	D_m	Pu_1	…	Pu_n	c_1	…	C_n
	d_1	…	D_m	Pu_1	…	Pu_n	c_1	…	C_n
$\begin{cases} p_{ri,1} \\ p_{ri,2} \\ p_{ri,r} \end{cases}$	d_1	…	D_m	Pu_1	…	Pu_n	C_1	…	C_n
	d_1	…	D_m	Pu_1	…	Pu_n	C_1	…	C_n
	d_1	…	D_m	Pu_1	…	Pu_n	C_1	…	C_n
$\begin{cases} p_{rN,1} \\ p_{ri,2} \\ p_{rN,r} \end{cases}$	d_1	…	D_m	Pu_1		Pu_n	C_1	…	C_n
	d_1	…	D_m	Pu_1		Pu_n	C_1	…	C_n
	d_1	…	D_m	Pu_1		Pu_n	C_1	…	C_n

Thus, to realize the context representation model, the structure is represented as a three-layer structure consisting of low-level sensor data and high-level contextual information, which can be related to each other to derive higher-level semantic contextual information, as described in chapter three. The user preference model is then formally defined in a general form as follows.

$$Y = f(x_1, x_2, x_3, \ldots, x_n) \tag{4.2}$$

Where $x_1, x_2, x_3, \ldots, x_n$ are the demographic attributes from A and the preference information from P_{ui} characterized by set G, C_a, and Ga. Y is the dependent variable to be determined , and function f is the predictive function learned via some classification algorithm. For example, the system could predict the media item that user u_i would like to consume using this model. However, the model defined in equation (4.2) does not account for contextual preferences of users. To incorporate contextual information into the model, we modify the above formal definition as follows.

$$Y = f(x_1 c_1, x_2 c_2, x_3 c_3, \ldots, x_n c_n) \text{ or } Y = fc_n(x_1, x_2, x_3, \ldots, x_n) \tag{4.3}$$

Where $c_i, \ldots, c_n$ represent the contextual information, and f_{cn} is the function responsible for identifying users' contextual preferences. This model can be used to build contextual user profile, and can be executed in non-contextual mode to determine user preferences without using contextual information. This means that the contextual user profile can operate in two modes: the contextual and traditional modes. The traditional mode is executed in situation here user's contextual situation is not available. In this mode, an entire profile considering user consumption history (if any is) used. Alternatively, it can use the consumption history of users who are similar to the current user to build the profile. The default mode is the contextual mode in which user's contextual information is used to determine user's preferences.

4.4.2 Obtaining contextual user preferences

To provide a mobile user with relevant media items, inferring his contextual preferences is a crucial process. The user preferences are identified based on the contextual situation of the user to infer the current preference accurately. To achieve this, we adopted a case-based methodology approach to retrieve and to learn contextual user preferences based on the currently identified user context. Please note that there are other methods such as machine learning based data mining processes such as Bayesian algorithm, Neural network etc. as discussed in chapter two. In fact, the design and implementation of the CAMR developed in chapter five allows incorporation of other user preference learning algorithms via a factory design pattern. The following are the steps to obtain the contextual preferences of a target user.

(a) Case-based methodology for identifying user preferences

Case-based reasoning or CBR is a machine learning method that solves a new problem, remembering a previous similar situation, by reusing information and knowledge in that situation [83]. It relies on the general knowledge of the problem domain or making association along generalized relationships between problem description and conclusion. It uses the specific knowledge of previously experienced, concrete problem situation (cases) to solve a newly identified problem. The new problem is solved by finding a similar past case and reusing it in the new problem situation. One unique and important feature of CBR is its ability for incremental and sustained learning from a new experience, which is retained each time a problem is solved, making it immediately available for future problems [83].

In a context-aware media recommendation system, CBR can be used to learn the contextual user profile, using it to retrieve past user's contextual preferences in the user profile model, or contextual preferences of other users in the knowledge base, similar to the currently recognized user's contextual situation [86].

To accomplish this objective, four major processes corresponding to the processes, making up the CBR cycles have been developed. These are cyclic processes as earlier introduced and illustrated in Figure 4.4. CBR uses k-nearest neighbor algorithm for retrieving similar cases. In order to compare the existing k-contextual preferences and current

contextual preferences, CBR naturally supports similarity function, to compute the distance between those preferences.

(b) Identifying and retrieving the current contextual preference

Obtaining the current contextual preference by comparing present contextual situation and past contextual situations (with associated preferences), in CBR, is similar to solving a new problem case by using solutions to past problem cases. Previously identified contextual situations and associated preferences constitute the best case, whereas a newly identified contextual situation is the new case. Therefore, a contextual preference case is described by a tuple <premise, value>. The premise is the description of the case with its characteristics, while value is the result of reasoning, based on the premise. Formally, the *premise* is a contextual situation (C) of a mobile user when consuming multimedia content, whereas *value* is the user preference ($Pi^i_{,c}$) in that situation. A contextual situation (a case) is represented as $Cs = \{(C^i, Pi^i_{,c})\}$, where C^i and $Pi^i_{,c}$ are contexts and associated preferences respectively. One of the preferences corresponds to the newly identified contextual situation. To select the most relevant contextual preference for the present contextual situation in the recommendation process, the system compares the present and the past contextual situations stored in the contextual user profile knowledge base. For example, intuitively, a contextual preference could be retrieved from the contextual user profile, as "*On rainy weekends, when she is at home, Ana prefers to watch romance movies*". In this example, the following contextual information can be identified: *User = Ana, Content category = Movie, Content Genre = Romance, Location = Home, Weather = raining, Time = Evening, Day = Weekend*. The currently recognized contextual situation is obtained from the context-awareness framework and represented in a vector Cs = (c_l^*,c_w^*,c_t^*,c_a^*,c_i^*,c_n^*), where c_l^*,c_w^*,c_t^*,c_a^*,c_i^*,c_n^* represents a user location, days of the week, time of the day, user activity, illumination, noise level of the location, etc. for instance. This is the basic context information, which is used to determine user preferences. For a user vector, provided as input to the recommendation process, a current contextual situation (case) defined as CC* = (C*,?) is built by the profile manager to determine its value part in the next process, where C* is the context.

(c) Retrieving the most similar contextual situations and preference

To determine the contextual preference in the present contextual situation case (CC*), the present case(context) is compared to the past contextual situations represented as vector $PC_S = \{C^l, C^n\}$, we select the contextual case that verifies the distance function in equation (4.4) similar to Bouidghaghen et al. [77] and Lee & Lee [85]. Where P_{ni} is the weighted intensity, representing the dynamic contextual preference of the user and value associated with the terms in the user profile. $f(C_i^l, C_i^n)$ is the similarity metric for comparing C_i^l *and* C_i^n, which are the values of the feature *i* in the contextual situation vectors respectively.

$$C^{opt} = \text{argmax} \ \frac{\sum_{i=1}^{n} Pi,cf(c_i^l, c_i^n)}{\sum_i^n p_{i,c}} \tag{4.4}$$

In order to be able to compare contextual features, we classify them into two categories as shown in Table 4.2. First, the categorical context features which consist of unordered values, e.g. activity contexts are examples with values such as *Walking, Running, Jogging*, etc. Second, the numeric context feature, which consists of ordered values, examples are illumination, temperature, etc.

. (1) To obtain the similarity between two numeric contextual preference cases, we used the equation (4.5).

$$f(C_i^l, C_i^n) = \begin{cases} 1 - \text{p if } 0 \le p \le 1 \\ 0 \quad \text{if} \quad \text{p} > 1 \end{cases} \tag{4.5}$$

Where $p = \frac{|c_i^l - c_i^n|}{Max - Min}$, Max is the maximum value of the i_{th} features of all cases in the case base and Min is the minimum value of the i_{th} feature for all cases in the case base. Let us take noise level as an example, if min is 10 dB, max = 150dB, current, x is 50, and y = 60 then the value of p is 0.0714. In this case, the similarity between x and y is 0.93.

Table 4.2- Example context information in a typical user profile preferences

Context	Values	Type
Location	*Home, Office, Mall, School, Coffee shop, Train station, Bus station*	*Categorical*
Weekday	*Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday*	*Categorical*
Weekend	*Saturday, Sunday*	*Categorical*
Holiday	*Weekend, others*	*Categorical*
Time of the Day	*Morning, Afternoon, Evening, Night*	*Categorical*
Weather	*Cloudy, Rainy, Snowing, Cloudy, Mostly Cloudy, Partly Cloudy*	*Categorical*
Current Temperature	*Very High, Normal, Low, Very Low,*	*Numeric*
Activity	*Walking, Sitting, Running, Jogging, Driving, Standing, ascending stairs, descending stairs, ascending elevator, and descending elevator.*	*Categorical*
Media Category	*News, Music, Movie*	*Categorical*
Genre	*Action, Animation, Comedy, Drama, Documentary, Epic, Horror, Politics, Sci Sports, Thriller, etc.*	*Categorical*
Language	*English, French, German, Italian, Portuguese, Spanish, etc.*	*Categorical*
Duration	*Short (<30 min), Medium (>30 min, <75 min), Long (>75 min)*	*Numeric*
Country	*Canada, France, Germany, Italy, Portugal, Brazil, Spain, UK, USA, etc.*	*Categorical*
Date	*Old (>5 years), recent (<5 years, >1 years), present (<1 year)*	*Numeric*
Illumination	*Dark, Very dark, Moderately dark, Bright, Moderately bright , Very bright*	*Numeric*
Noise level	*Quiet, moderately quiet, Very quiet, Loud, Moderately loud, Very loud*	*Numeric*

```
Get new contextual situation
        Set preference = null
        Set similarity threshold
        NumberOfSimilarContext = 0
        NumberOfCases = 0
    For each contextual case
    For EachContext, match = false

                Compute Similarity between the new context and contextual situation x using equation (4.5) or
                using predefined similarity scores
                        If similarity ≥ threshold
                                Increment NumberOfSimilarContext
                                Increment NumberOfCases
                                Match = true
                                Preference = preference (retrieve corresponding preference)
                        EndIf
                        ElseIf
                                Break
                EndFor
                If NumberOfCases = TotalNumberOfCases
                        Preference = getPreference
                        Break
                EndIf
EndFor
```

Figure 4.7- Retrieving similar contexts from user profile

(2) To obtain similarity between two categorical features, a partial matching scheme similar to Lee & Lee [85] was developed using the domain knowledge. Table 4.3 presents the similarity scores for some select categorical features from the system' knowledge base used in our proposed solution. Figure 4.7 shows the heuristic steps which addresses equation (4.4), using the similarity scores to obtain similar contextual situations as well as corresponding consumption preferences in those contexts.

(d) Reusing contextual situation to obtain the contextual user preference

To determine the contextual preference in the present contextual situation, the present context is compared to the existing contextual situations as discussed in section 4.4.5, similar to Bouidghaghen et al. [86] and Lee & Lee [85]. The value of the recognized current context of usage is compared to the leaves of the profile tree (context nodes) to identify the upper nodes that provide the values of the user's contextual preference. The process for comparing and obtaining a match between the existing contexts of the current user (if any) or of other users and currently identified context has been described in c. Only nodes whose leaves match the current context are processed by retrieving its intensity into a contextual user preference. Each user preference is a pair keyword-intensity. Keyword is the textual value of nodes (the type field); intensity is obtained by multiplying the values of the field's weight and lifetime of the respective node. The weight represents the preference of the user while lifetime represents the time elapse since the last time user consumes the content characterized by that *category-genre-property* node. The intensity (the dynamic preference of the user) of those elements belonging to the media's term in the *genre-property* level is computed by summing up the products [weight x lifetime] of all their occurrences using equation (4.1) as explained in section 4.4.1.1. The intensity of the retained elements at this level is obtained by visiting their child nodes in a breadth-first traversal. The same applies to the retained elements of the category level.

Table 4.3- Example similarity scores for categorical context features

Contexs	Categorical similarity scores							
Activity	New Case / Old Case	Walking	Running	Jogging	Driving	Lying	Standing	Sitting
	Walking	1.0	0.2	0.3	0.0	0.0	0.0	0.0
	Running	0.2	1.0	0.5	0.3	0.0	0.2	0.0
	Jogging	0.3	0.5	1.0	0.1	0.0	0.1	0.0
	Driving	0.0	0.3	0.1	1.0	0.0	0.0	0.0
	Lying	0.0	0.0	0.0	0.0	1.0	0.2	0.5
	Standing	0.1	0.0	0.0	0.0	0.2	1.0	0.3
	Sitting	0.1	0.0	0.0	0.0	0.5	0.3	1.0
…	…	…	…	…	…	…	…	…
Days		Weekday	Weekend	Holidays				
	Weekdays	1.0	0.1	0.2				
	Weekend	0.1	1.0	0.9				
	Holidays	0.1	0.9	1.0				
Time								
		Morning	Afternoon	Evening	Night			
	Morning	1.0	0.3	0.1	0.0			
	Afternoon	0.3	1.0	0.5	0.3			
	Evening	0.1	0.5	1.0	0.5			
	Night	0.0	0.3	0.5	1.0			
…	…	…	…	…	…	…	…	…

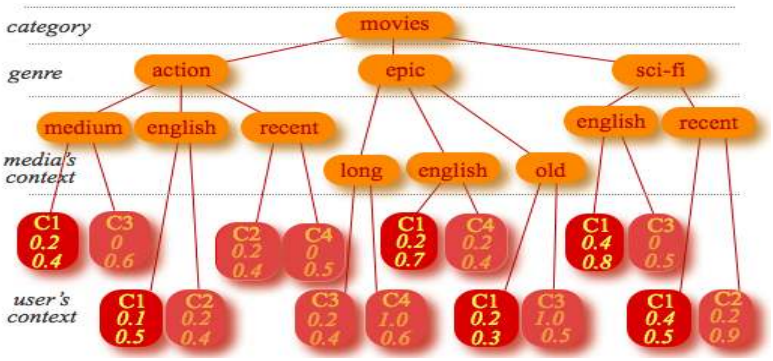


Figure 4.8- Example user profile representation for movies.

V_c = *[(medium,0.08),(English,0.51),(old,0.06),(recent,0.2),(action,0.08),(epic,0.14),(sci-fi,0.32),(movies,0.54)]*

Figure 4.9- Hypothetical user profile vector.

The intensity of the elements belonging to the genre level is the largest value of their children. Accordingly, these values are obtained by performing a depth-first traversal. As an example, consider a hypothetical user profile of Mr. X represented in Figure 4.8. Assuming that the system has inferred that Mr. X is in context C_1, the elements that will be included in the contextualized user profile vector are the ones that have leaves with context value C_1. The intensity of those elements belonging to the media's term is computed by summing up the products [weight x lifetime] of all their occurrences (e.g., the node with value "English" occurs three times for context C_1; therefore the intensity of "English" is the sum of the corresponding three intensities:(0.1 x 0.5 + 0.2 x 0.7 + 0.4 x 0.8 = 0.51). Figure 4.9 represents the resulting contextualized user profile content preferences.

(e) Updating the contextual user profile (Revising and retaining contextual user preference)

To keep the system up to date on the continuously changing user's preferences, the framework tracks the user's consumption behavior, and accordingly updates the user profile to improve the system's classification accuracy. The user profile can be updated using two approaches, namely explicit and the implicit methods [10], [46], [56], [130]. The former grants the users' ability to modify their dynamic contextual preferences by the system. This gives users some level of control where they can express directly how they feel about the recommendations they receive. The implicit approach involves preference learning without direct user intervention, such as updating the profile when the user has spent a certain amount of time visualizing a given recommended content. The profile is updated whenever the user consumes any kind of media item. This allows associating importance to the finer detail of content, as well as the corresponding contexts of consumption.

The profile model associates weight $\in$ [0, 1] that relates the relevance of a provided media content preference to the contextual situation in which it is consumed, learning the user preference in the contextual situation.

This is achieved in two simple steps depending on the similarity obtained between current user contextual situation and the most similar situations.

(1) If the $C^* \neq C^{opt}$: Means a new contextual situation (case) has been discovered and this is added accordingly to the contextual user profile

(2) $C^* = C^{opt}$ The contextual situation is updated in the user profile to reflect the user preference.

An intensity w_{ij} represents the relevance of content with the preference p_{ri} belonging to the contextual situation CC^* that user u_i consumes in context C_i.

Whenever the context awareness model detects that the user is in a context C_i, for a continuous period of time [0, T] and the reasoner finds the preference of content $m_1, m_2, \ldots m_n$, whose contextual situations match the present contextual situation of the user, then weights $w_ic_i \in [0,1]$ is associated with this content preference, which at time T is updated as follows:

$$w_ic_i = w_{i-1}c_{i-1} + \gamma(\alpha - w_{i-1}c_{i-1})\ i = 1,2,3,\ldots,n. \quad (4.7)$$

Then for those content b_1, b_2, $b_3,\ldots,b_n$ with associated weights w_1b_1, w_2b_2, $w_3b_3,\ldots,w_nb_n$ not consumed by the user, these weights are updated as follows:

$$w_ic_i = w_{i-1}c_{i-1} - \gamma(\alpha - w_{i-1}c_{i-1})\ i = 1,2,3,\ldots,n. \quad (4.8)$$

γ is a learning parameter whose value is obtained by:

$$\gamma = 1 - \left(\frac{t}{45}\right)^5 \quad (4.9)$$

$\alpha \in [0,1]$ its value is set to 1 in (4.7) and 0 in (4.8)

Factor t in formula (4.9) represents the number of days elapsed since the last time the user has consumed an item with the characteristics described by his profile nodes. With equation (4.9), the parameter γ for each node remains above 0.9 during the first 30 days after it has been visited, rapidly decreasing to zero after that period. For all other nodes, γ is linearly increased daily. This way, nodes in the user profile that represent items that have not been seen for a long period, will eventually have no impact on the user preferences evaluation. Note: value 45 in equation 4.9 was used as a "good guess" but further study is required to fine-tune its value.

4.5 Context-aware personalized media recommendations

To provide personalized multimedia content of different categories to mobile users, the proposed a conceptual context aware recommendation framework incorporating three traditional recommendation techniques with capabilities to provide suggestions based on context-aware content-based recommendations, context-aware collaborative-based recommendations and context-aware hybrid recommendations. The context-aware content-based recommendation was developed to use contextual information to recommend a set of candidate media items to users. Recommendation is generated according to the user's current context and content she has consumed in similar contexts in the past, designated as context-aware content-based recommendation. Context-aware collaborative process recommends a set of media items to a user according to the relevance of content consumed by likeminded users in contexts similar to the target user's current context, using the contextual profiles of those users, as well as the target user's contextual profile and the media item metadata. We also proposed a combination of the two approaches as a context-aware hybrid recommendation process, where contextual situations in which old users have consumed media items in the past to provide contextual recommendations to the user with little or no contextual history. In the next sections, we introduce these processes, as extensions of traditional recommendation techniques based on the contextual user profile model presented in the last section.

4.5.1 Context-aware content-based recommendations

The traditional content-based media recommendation systems capture the relationships between multimedia items and a user's preference, using media descriptions and user profile as illustrated in Figure 4.10. The user profile represents a user's interests in specific multimedia items. The user profile information (i.e. current user) and the media item descriptions serve as the inputs to the recommendation process. In addition, it determines the similarity between the user's interest profile and multimedia content by matching up the multimedia descriptions, and user profiles to produce a similarity matrix. Initially, users are asked to fill in the profile with some interesting information, which is used to

provide recommendations. However, this information is not always reliable because the information supplied by the user always changes according to their dynamic interests.

This traditional recommendation system approach cannot effectively provide relevant and quality recommendations in a mobile environment because the information supplied by the user does not always capture their interests based on their contexts, and therefore cannot be confidently relied upon. To address this limitation, we propose an extension of traditional content-based recommendation process, with contextual information as shown in Figure 4.11. The extension allows the activation of the context-free recommendation part of the process in situation where contextual information of the user is not available.

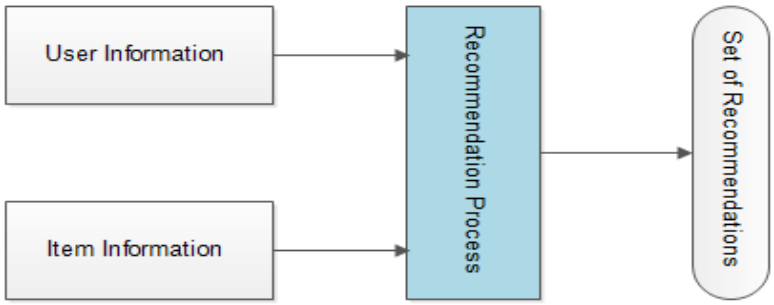


Figure 4.10-A traditional recommendation process

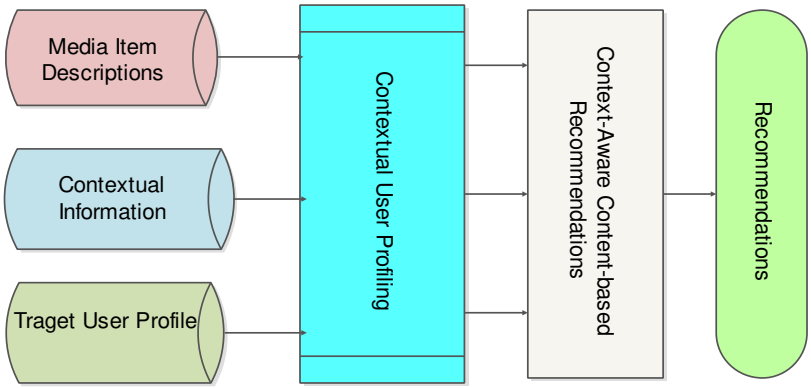


Figure 4.11- A generic context-aware content-based recommendation process

Specifically, this means that we could use context-aware content based recommendation, as well as the traditional content-based recommendation, in conjunction with the contextual user profile model by suggesting multimedia items to users based on the contexts in which they have consumed content in the past as well as the descriptions of those consumed content.

4.5.1.1 Context-aware vector space model (VSM) for content-based recommendations

Vector space model (VSM) is generally a simple algebraic model representing textual information as a vector [7]. Used extensively in information filtering and retrieval, ranking and indexing, VSM could be used to match user profile and candidate media item profile in order to easily obtain similarity between them. The basic reason for adopting and for using this model is that various candidate media items and user information can be represented as elements of user and media item vectors. VSM was developed to address the representations of media item content and user information, and the difficulties of determining the relevance of each candidate media item to an active user. It allows us to compute the continuous degree of similarity between contextual user profiles and media items, and subsequently allowing ranking of media items according to their degree of relevance or similarity to the user profiles.

It also allows the flexibility to compute the continuous similarities between users or between media items. For example, it allows developing item based or user based recommendation processes much more easily.
We could determine, using VSM, users that could be regarded as friends (neighbors), based on user-to-user similarities. Similarly, it also affords the flexibility to determine media items that are similar to each other that is item-to-item similarities. Above all, it allows the flexibility to include other information such as contextual information intuitively in addition to user and item information.

In this regard, in order to identify likely candidate media items a target user would prefer in a given contextual situation, some method of representation of media items must be provided. In this thesis, the descriptions of the candidate media items are assumed to have been represented in MPEG-7 MDS descriptors. From MPEG-7 descriptors, specific key terms describing the media items can be extracted as linear elements of vectors, each vector representing a media item.

Assuming we have a set of candidate multimedia content, let *t* be the number of key terms, which have been extracted from the descriptors, describing each media item:

$K = \{k_1, \ldots, k_t\}$ is the set of all terms describing media items belonging to a given category. A weight $w_{i,j} \geq 0$ is associated with each term of k_i of a media item m_j. Every term, which does not appear in the description of a specific media item, $w_{i,j} = 0$. In the contextual user profiles, $w_{i,j}$ represents the contextual user preference, i.e. the intensity $p_{ni,}$ given in equation (4.1)

Therefore, each media item is associated to a media term vector $\overrightarrow{V_m}$

$$\overrightarrow{V_m} = (w_{1,m}, w_{2,m}, \ldots, w_{t,m}) \tag{4.10}$$

In addition, a set of mobile users whose contextual preferences are represented by:

$$\overrightarrow{V_c} = (p_{1,c}, p_{2,c}, \ldots, p_{t,c}) \tag{4.11}$$

In information retrieval, $\overrightarrow{V_m}$and $\overrightarrow{V_c}$ are documents and queries respectively. However, in the proposed framework, these represent candidate media items and contextual user preferences. The question, therefore, remains of how to obtain $w_{1,m}, w_{2,m}, \ldots, w_{t,m}$ and $p_{1,c}, p_{2,c}, \ldots, p_{t,c}$ representations of media items and user preferences, respectively from the contextual user profile. In information retrieval, there are several methods, called scoring, which are used to obtain these weights. Two popular functions in information retrieval are the weighted zone scoring and term frequency scoring [7]. The frequency scoring, in particular, relies on three factors. The frequency, a collection of frequency, and length normalization factors. These factors are multiplied to obtain the resulting term weight. However, we did not use these methods. Instead, the scores are obtained using the contextual user profiles in form of intensity values, which represent the user's dynamic contextual preferences. However, to use vector space model, we represent this contextual preferences and media items in vectors $\overrightarrow{V_c}$ and $\overrightarrow{V_m}$ respectively. For example, to obtain the representation of information about users and media items in vector form, the proposed system devised a method to retrieve dynamic contextual preferences of the users from the user profile into a vector representing the each user as described in section 4.4. Similarly, this contextual information vector is compared with the media item descriptions to obtain the vectors representing the candidate media items described in section 4.5.1.3.

4.5.1.2 Contextual user profile vector

The representation of an active user's contextual preference is a vector, $\overrightarrow{V_c}$, obtained from the contextual user profile. A global version of this user vector has as many elements as the number of different *category-genre-properties* that exist in the complete user profile structure. A contextual user vector typically has a much smaller number of elements, corresponding to different category-genre-properties associated with specific contexts under consideration.

The contextual user vector $\overrightarrow{V_c}$ is built using contextual data to filter the user profile to determine the user's current preference and then to obtain the respective elements of $\overrightarrow{V_c}$, by comparing the user' current contextual information with the leaves of the profile tree (context nodes) to identify the upper nodes that provide values for the elements of $\overrightarrow{V_c}$. Only nodes whose leaves match the current context are selected and retrieved. Using the active user's current contexts to retrieve similar or matching contexts from the user profile repository is a complex process, which is addressed by the processes described in sections 4.4.1 and 4.4.2. Each element in the vector is a pair keyword-intensity. Keyword is the textual value of nodes (the type field); intensity is obtained by multiplying the values of the fields' weight and lifetime of the respective nodes.

4.5.1.3 Multimedia content profiles

The system classifies candidate media items as relevant to a user by first computing the elements of vector, $\overrightarrow{V_m}$, which it does by computing $\overrightarrow{V_m}$'s similarity to the contextual user profile vector $\overrightarrow{V_c}$. It is therefore necessary to create a media vector, $\overrightarrow{V_m}$, for each candidate media item. To obtain the description of candidate media items, the proposed CAMR framework relies on the availability of semantic metadata using the MPEG-7 MDS semantic tools. Given that in practice most of the media resources available online lack this kind of metadata, the proposed system incorporates alternative methods for obtaining semantic descriptions of the candidate content. One of such alternatives is to query the Internet using existing open source API such as the Internet Movie Data Base (IMDB). For each media item, a vector $\overrightarrow{V_m}$ is initially created as an exact replica of $\overrightarrow{V_c}$. Then, for each element of $\overrightarrow{V_m}$, the system inspects the MPEG-7 metadata for a match. If it finds a match, it retains the intensity value of the matching element in $\overrightarrow{V_m}$. Otherwise, it allocates the value 0 (zero).

Figure 4.12 shows an example MPEG-7 metadata of a candidate media item. Figure 4.13 illustrates the corresponding media vector, built using the contextual user vector presented in Figure 4.9. This idea was adapted from [18] but implemented in a simpler way, without defining an additional vector of attributes and weights. Instead, the elements of $\overrightarrow{V_c}$ are used by default. The process for using $\overrightarrow{V_c}$ to obtain $\overrightarrow{V_m}$ is now explained. Note that the elements of $\overrightarrow{V_c}$ are retrieved with a reference to the keyword associated with it. A $\overrightarrow{V_m}$ therefore, is created with an exact number of elements as $\overrightarrow{V_c}$. Then, the keywords associated with the $\overrightarrow{V_c}$'s elements are compared with the content of the candidate media items' metadata. A match triggers the system to retrieve the corresponding element of $\overrightarrow{V_c}$. In addition, each of the keywords belongs to a category with different levels of importance, which represents the contribution they provide in the characterization of the content or the user preferences. For example, for the category "movies", the property "genre" is considered more relevant than, say, "*keywords*" or "*duration*". Therefore, when a match is found between the keyword associated with the elements of $\overrightarrow{V_c}$ and metadata of candidate items, the category of such item then is looked up and its importance value multiplied by the corresponding elements retrieved from $\overrightarrow{V_c}$.

```
<MPEG7>
...
<DescriptionMetadata>
<Genre>action</Genre>
<Language>English</Language>
<MediaDuration>PT0h55M30S</MediaDuration>
<CreationTime>2005-10-10T19:45:00+09:00</CreationTime>
<Textannotation>
...
</MPEG7>
```

Figure 4.12- Excerpt of hypothetical MPEG-7 metadata

4.5.1.4 Obtaining the similarity between a target user and candidate media item

Having obtained the information representing the user preferences and media items profiles described in the last processes in sections 4.5.1.2 and 4.5.1.3, the next process is to determine those candidate media items that match the target user's profile.

This is the process that leads to obtaining context-aware content based recommendations using one of the established similarity or distance methods. The purpose is to compute the distance between $\overrightarrow{V_c}$ and $\overrightarrow{V_m}$, this distance represents the similarity between the candidate media content and the active user. The output of this process is a user-media item preference matrix, which is used in the recommendation process. Cosine distance equation (4.12) can be used to obtain the similarity. For each item, a predicted dynamic preference can be then be obtained using equation (4.13).

$$Sim(\vec{v}_c, \vec{v}_m) = \frac{V_c . V_m}{V_c \times V_m} = \frac{\sum_{i=1}^{n} p_{i,c}\, w_{i,m}}{\sqrt{\sum_{i=1}^{n} p_{i,c}^2}\sqrt{\sum_{i=1}^{n} w_{i,m}^2}} \tag{4.12}$$

$$Pni = \frac{\sum_{i=1}^{n} Sim(\vec{v}_c, \vec{v}_m)\, w_{i,\mathrm{m}}}{\left|\sum_{i}^{n} w_{i,m}\right|} \tag{4.13}$$

The computed preferences for the user of each candidate media item are then ranked in descending order of magnitude. The *top k* items with the highest preferences are provided as recommendations. Figure 4.14 summarizes the contextual content-based recommendation process and Figure 4.15 illustrates the workflow of the context-aware content-based recommendation process respectively. Where pn_i is the weighted preference of item m for user c computed over items he has consumed in the past contextual situations. Because the similarity between the user profile and the closest candidate media items vary and this may affect the final preference prediction for the user, then we weighted the contribution of each candidate item to the final prediction, p_{ni}, using the denominator of equation (4.13). Where n is the number of candidate media items (most similar to the target user).

V_m = *[(medium,0.08),(English,0.51),(old,0),(recent,0.2),(action,0.08),(epic,0),(sci-fi,0),(movies,0.54)]*

Figure 4.13 - Hypothetical candidate media vector

Process: context-aware content-based recommendations
Input: current_user_contextual_profile, media profiles, contextual information
Output: A set of top n recommended media items

1)Sense and classify active user present context

2) Compute the similarity between the target user current context and his past contexts(if any) $\boldsymbol{f(C_i^l, C_i^n)}$

- ***a. Process target user profile to obtain his preference for the current context***
  - ***i. Retrieve media metadata from the web***
  - ***ii. Generate target user contextual profile vector***
  - ***iii. Generate media content profile vectors***
  - ***iv. Compute the similarity between candidate media content and target user vectors (4.12)***
  - ***v. Rank the TopN candidate content for the active user according to the similarity and predicted preferences(4.13)***
- ***b. If no match is found:***
  - ***vi. Retrieve media metadata from the Internet***
  - ***vii. Generate non-contextual user profile vector***
  - ***viii. Generate media content profile vectors repeat a,iv and a v***
- ***c. Present suggested items to active user***
  Record feedback(implicit/explicit)
- ***d. Log his present context and update his user profile***

Figure 4.14- Context-aware content-based recommendation process

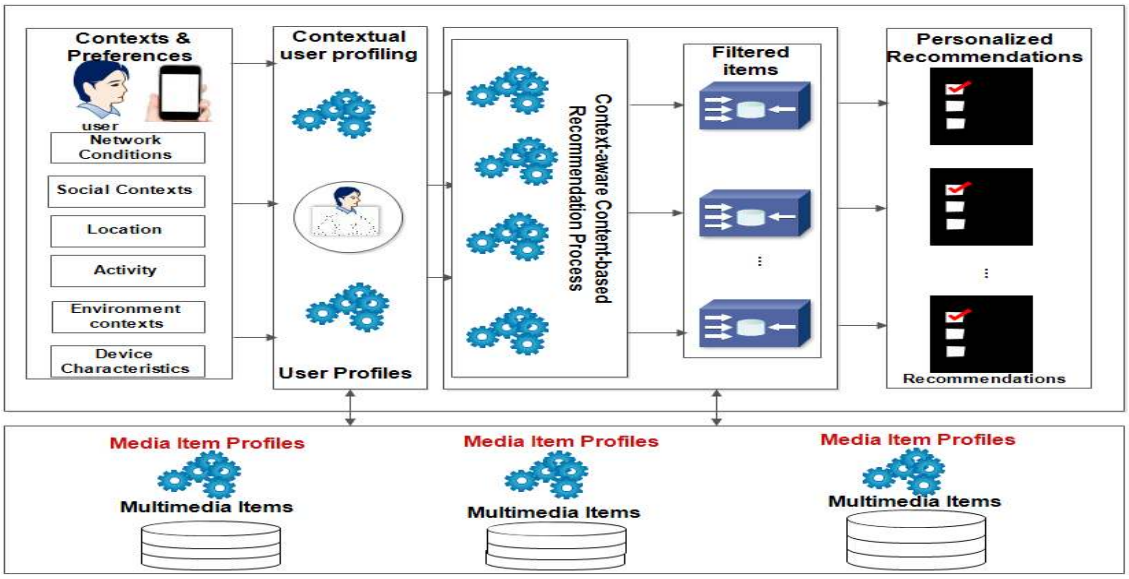


Figure 4.15- CAMR context-aware content-based recommendation process workflow

4.5.2 Context-aware collaborative recommendations

Context-aware content-based recommendations described in the last section suggests content to users based on their contextual consumption and descriptions of available candidate items. One of the benefits of that recommendation process is that it allows limited sharing of user information because it does not entail collecting information from other users. Nevertheless, one of its main weakness is that it suffers from the overspecialization problem [10], where such system could only recommend multimedia items that are similar to the ones the user has consumed and/or rated high in the past. Any multimedia items different from those are always excluded from the recommendation process. In addition, because a user must have rated a large number of items in the past, traditional recommendations including collaborative based filtering suffer the so-called new user problem also known as coldstart, where the system fails to recommend items to users who have rarely used the system or those users who are using the system for the first time. Another problem with the process is that features characterizing candidate items must be explicitly specified before such item could be considered in the recommendation process. These solutions are very complex, costly and are not always reliable. Conventional collaborative recommendation suggests content to users, based on the opinions and preferences of other like-minded users [15].

In the proposed CAMR framework, the context-aware collaborative-based process suggests content to a target user, based on her current context and on the contents that other like-minded users have consumed under similar contexts. Figure 4.16 shows the generic context-aware collaborative recommendation architecture and Figure 4.18 shows the recommendation steps. In Figure 4.17, basic workflow of the context-aware collaborative recommendation is depicted. The difference between workflow of context-aware content based recommendation illustrated in Figure 4.15 and that of context-aware collaborative recommendation is depicted by multiple user profiles in Figure 4.17 representing neighbors of the current user unlike in Figure 4.15 (context-aware content based) where only one user profile is involved in the recommendation process.

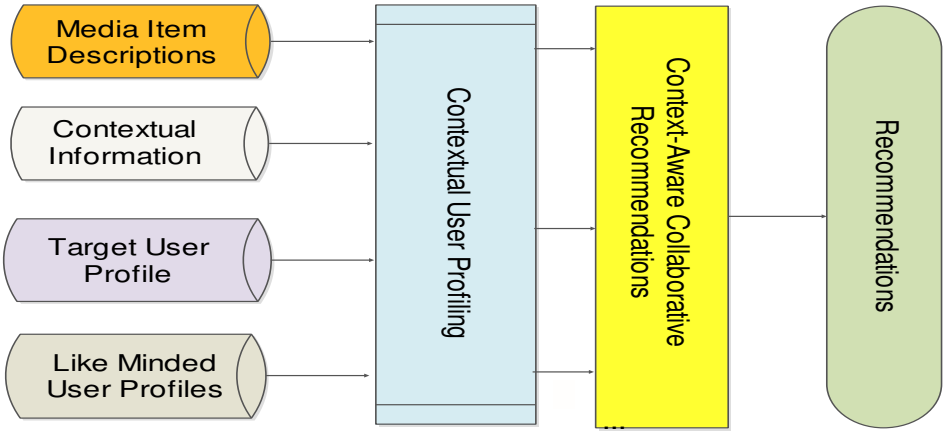


Figure 4.16- A generic context-aware collaborative filtering based recommendation .process.

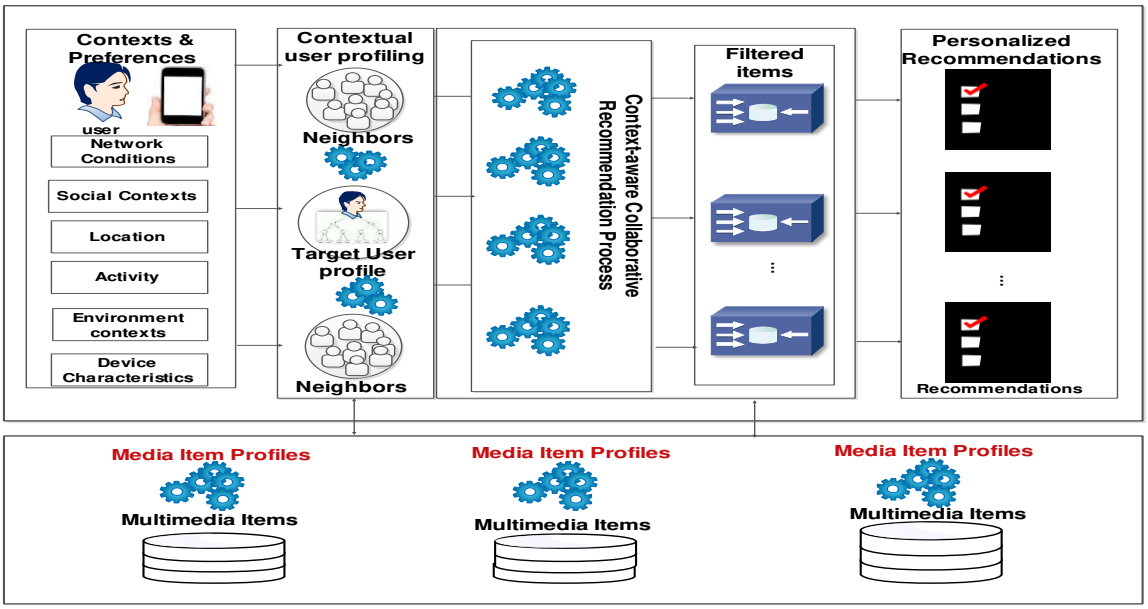


Figure 4.17- CAMR context-aware collaborative filtering process workflow.

The proposed system incorporates a context-aware collaborative recommendation process. Unlike the context-aware content-based process, it utilizes dynamic preferences of similar users in the recommendation process. We define similar users as those who have consumed content in the context similar to the context of the active user. The context-aware collaborative recommendation process operates in six main steps as follows, similar to the context-aware process described in [14].

(1) In the first process, the system identifies users who have consumed items in contexts similar to the target user's context as described in contextual user profiling process in section 4.2.2. The algorithm identifies every user (neighbor) that is similar to the active user looking for context that matches the target user's recognized context.

(2) In step 2, the system obtains the set k of users, u_j, with the highest context similarity from step (1). For every user profile with a match, the intensity value p_{ni} of the *category-genre-property* nodes in the user profiles is retrieved. To obtain the similarities between owners of those profiles with matching contexts and the target user, u_i, the *k-Nearest Neighbor* collaborative algorithm is employed by first obtaining the user preferences in such contexts. The values of

these preferences are used in equations 4.14 and 4.15 to obtain a set of users that we call neighbors or friends of the current users, i.e., a set of top *k* most similar users, U_K using the contextual information of the target user.

(3) In the third step, for individual neighbor $\mathrm{u}_\mathrm{i} \subset U_K$ among the k users of step 2, the content he/she consumes in contexts similar to that of the active user, u_n, is filtered, and the similarity between those consumed by each of the other k-1 neighbors (u_j) is computed using equation (4.14), which combines the preferences of this user u_j and every other neighbor $\mathrm{u_p}$ into a weighted average, using the correlations of these neighbors as the weights.

In (4.14) $p_{i,c}$ intensity (dynamic preference) of items with *category-genre-property* consumed by user u_i, $\overline{p_{i,c}}$ is average intensity for user u_i. $p_{j,c}$ is the intensity of items with *category-genre-property* consumed by user u_j in context c, and $\overline{p_{j,c}}$ is the average intensity.

$$p_{ni}(u_i, u_j, c \in C) == \frac{\sum_{c \in C}(p_{i,c} - \overline{p_{i,c}})(p_{j,c} - \overline{p_{j,c}})}{\sqrt{\sum_{c \in C}(p_{i,c} - \overline{p_{i,c}})^2}\sqrt{\sum_{c \in C}(p_{j,c} - \overline{p_{j,c}})^2}} \tag{4.14}$$

Where $c \subset C$ summations over media items containing category-genre-property that both user u_i and u_j have consumed in similar context c.

(4) In the fourth step, we define a weighted preference ($p_{nu_c,i,c}$) (4.15) for each neighbor u_j for an item consumed in context c , similar to the current context c_n of the new user, using context similarity $f(C_i^l, C_i^n)$ as weights in (4.15). Based on category-genre-property of items consumed by each neighbor, this step generates weighted preferences for each neighbor using their context similarity and preferences stored in each profile for each category-genre-property. Because the consumption context (c or c_n) is multidimensional, e.g. each location of the user is characterized by e.g. the activity the user engages in that location, the location illumination, the time of the day at the location, the day of the week, etc. the similarity of these context dimensions is computed and summed as illustrated in equation (4.15). m in (4.15) is a normalizing factor such that the absolute values of the context similarity sum to unity.

$$p_{nu_c,i,c} = m \sum_{c \in C} \sum_{j=1}^{k} p_{ni}(u_i, u_j, c \in C).f(C_i^l, C_i^n) \tag{4.15}$$

Note: $c \equiv C_i^l$ the contexts in which neighbors $\mathrm{u_j} \subset U_K$ have consumed media items c_n is the same as C_i^n

(5) In this step, each category-genre-property with the highest value obtained from (4.15) is then used to obtain content with similar category-genre-property features from online sources.

(6) In the final step, the similarity between category-genre-property features of items obtained in step 4 is used to compute the predicted relevance, p_{u_n,i_n,c_n}, of each candidate item using equation (4.16), ranked in decreasing order of magnitude. The first n items with the highest values are presented to the user as a set of recommendations. Figure 4.18 summarizes the entire context-aware collaborative recommendation process.

$$p_{u_n,i_n,c_n} = \overline{p_{i,c}} + m \sum_{u=1}^{k} \left(p_{nu_c,i,c} - \overline{p_{j,c}} \right). p_{ni}(u_i, u_j, c \in C) \tag{4.16}$$

Process: Context-aware collaborative recommendations

1) ***Inputs: contextual information, contextual user profiles, media item profiles***
2) ***Output: a set of contextual recommendations***
3) ***Sense and classify target user's current context***
4) ***Compute the similarity between target user's current context and other users profiles***

   a. ***If context similarity*** $f(C_i^l, C_i^n)$
      i. ***Rank a set of topN users (neighbors) with the highest contextual similarity in (5) according to the similarity of their contexts to that of the target user.***
      ii. ***For each neighbor, retrieve (dynamic) preference based on content consumed in context(similar to target user current content)***
      iii. ***Compute a weighted preference*** $(p_{u_k,i_n,c})$ $using$ ***(4.15) for each neighbor*** u_k
      iv. ***Retrieve media metadata from the Web based on computed weighted preference in(iii)***
      v. ***For every content consumed this context by every neighbor***
         -Generate a prediction for the target user and produce a set of topN multimedia items as recommendations
   b. ***If no match is found:***
      i. ***Retrieve media metadata from the Internet***
      ii. ***Generate non-contextual user preference***
      iii. ***Generate media content profile vector repeat a,iv***
      iv. ***Using traditional CF generate a set of multimedia item recommendations***
   c. ***Present suggested items to active user***
   d. ***Obtain user feedback(implicit/explicit)***
   e. ***Log present context and update user profile***

Figure 4.18- Context-aware collaborative process

Table 4.4- User contextual consumption log showing new user problem — Consumption Records

User	High-level Contexts	Item1	Item2	Item3	Item4	Item5	…	Item n
User 1	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Sitting),(Location:Cinema),(Illumination: Bright),(Noise Level: Normal)} …	x	√	x	x	x		x
User 2	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location: Home),(Illumination: Bright),(Noise Level: Normal)} ….	√	x	x	x	x		x
User 3	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Jogging),(Location:Sport Complex),(Illumination: Bright),(Noise Level: Normal)} ….	x	√	x	x	x		x
User 4	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Sitting),(Location :Office),(Illumination: Bright),(Noise Level: Normal)} ….	√	x	x	x	x		x
User 5	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location: Home),(Illumination: Bright),(Noise Level: Normal)} …	x	x	x	√	x		x
User 1	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location: Home),(Illumination: Bright),(Noise Level: Normal)} …	√	x	x	x	x		x
User 2	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Sitting),(Location :Home),(Illumination: Bright), (Noise Level: Normal)} …	x	√	x	x	x		X
User 3	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location: Home),(Illumination: Bright),(Noise Level: Normal)} …	x	x	√	x	x		X
User 4	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location: Home),(Illumination: Bright),(Noise Level: Normal)} …	x	x	x	√	x		X
User 5	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Sitting),(Location :Home),(Illumination: Bright),(Noise Level: Normal)} …	√	x	x	x	x		X
User 6	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location: Home),(Illumination: Bright),(Noise Level: Normal)}	?	?	?	?	?		?

4.5.3 Context-aware hybrid recommendations

The traditional collaborative recommendation systems compute predictions for the current user based on the item rated or viewed by other users in the past, whereas content-based approach uses the consumption history of the user. These two approaches suffer from the so-called new user problem, which excludes casual or new users or those with limited

information from getting relevant recommendations [6]. In fact, context-aware recommendations naturally do not address the new user problem adequately, as illustrated in Table 4.4. In this table, a sample of context logs of some users shows that user number 6 is a user with no consumption history, since user 6 in the table appears to be using the system for the first time; she does not have adequate information. The only information we have about this user is her current context. In this situation, either collaborative or content-based approaches would fail to provide him with relevant media items. Hybrid recommenders combine one or more of the conventional recommendation processes to minimize the impact of the weaknesses of its constituent algorithms to gain better performance. To profit from the hybrid technique, the thesis developed a context-aware hybrid technique that combines the CF and CBF in a context-aware recommendation process.

The recommendation service explores the two recommendation algorithms for contextual hybrid multimedia item recommendations, combining the context-aware content-based filtering (CF) and the collaborative-based filtering (CBF) described in sections 4.5.4 and 4.5.5. For example, in Table 4.4, the proposed framework would provide recommendations for first or casual users using contextual information of users who have consumed content using the system in the past. The primary task in this situation is first to recognize the context of the new user and then finding those users who have consumed content in contexts similar to the newly identified contexts. Using the similarity between the contextual information, it obtains the preferences of those users for content consumed in these similar contexts, and then predicts the new user's likely preference in the current context. In this section, we introduce the hybrid process, namely a context-aware content-based collaborative recommendation algorithm, which combines the recommendation processes described in the last two sections. The hybrid process can be described as a feature augmentation hybridization, in which new features, namely, preferences, are generated using the first recommendation algorithm, and the newly generated features are used as inputs in the second recommendation process [28]. The content boosted collaborative filtering inspires this hybridization type, which uses content-based filtering predictor to enhance existing user data, and then provides personalized recommendations, using collaborative filtering [181].

However, unlike the content boosted collaborative recommendation algorithm, where the main recommendation algorithm is the collaborative recommendation, the proposed hybrid process uses contextual information in a collaborative algorithm to address the new user of traditional recommendations.
The fundamental principle of this process is first to identify the context of the current user and second to locate those users who have consumed media items in contexts that are similar to the identified context of the current user. Third, it determines those items that are similar to the items consumed by those users in such contexts. In other words, it works on the basis that items similar to those consumed by users in contexts similar to the target user's context will interest the target user.

Figure 4.19 depicts the generic architecture of the context-aware hybrid recommendation process and Figure 4.20 is the workflow of the proposed hybrid recommendation process. Note the difference between Figure 4.20, Figure 4.15 and Figure 4.17. The workflow illustrated in Figure 4.20 shows a hybrid process combining features of content based and collaborative-based processes. The hybrid process is realized in three phases summarized as follows

(1) In the first process, the system uses the contexts in which other users have consumed media items in the past to find *n* users that are similar to the target user. This first process is the same as the first process of the context-aware collaborative process described in the last section.

(2) In the second phase, the system uses the context-aware content-based recommendation to filter the profile of each neighbor of the target user i.e. the outputs (i.e. profiles of current user's neighbors) of the last steps as inputs of a context-aware content-based filter.

It then obtains and ranks the media items previously consumed by each neighbor(k most similar users), by computing vectors $\overrightarrow{V_c}$ and $\overrightarrow{V_m}$, corresponding to contextual user profile of each neighbor and candidate media profile vectors respectively of the consumed content.

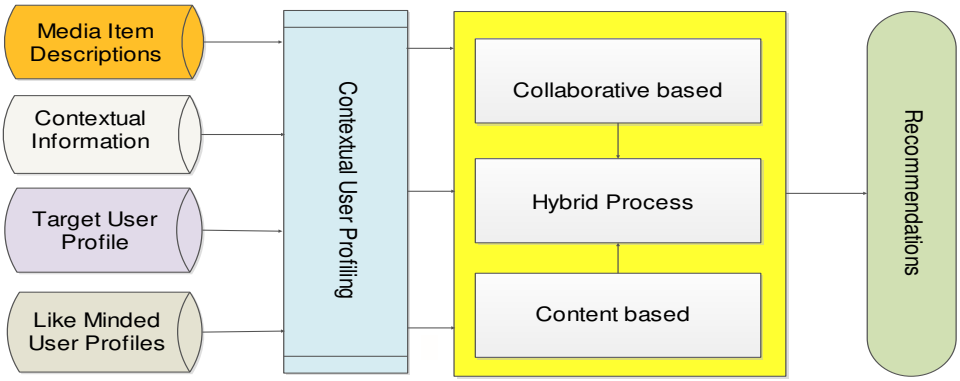


Figure 4.19- A generic architecture for context-aware hybrid recommendation

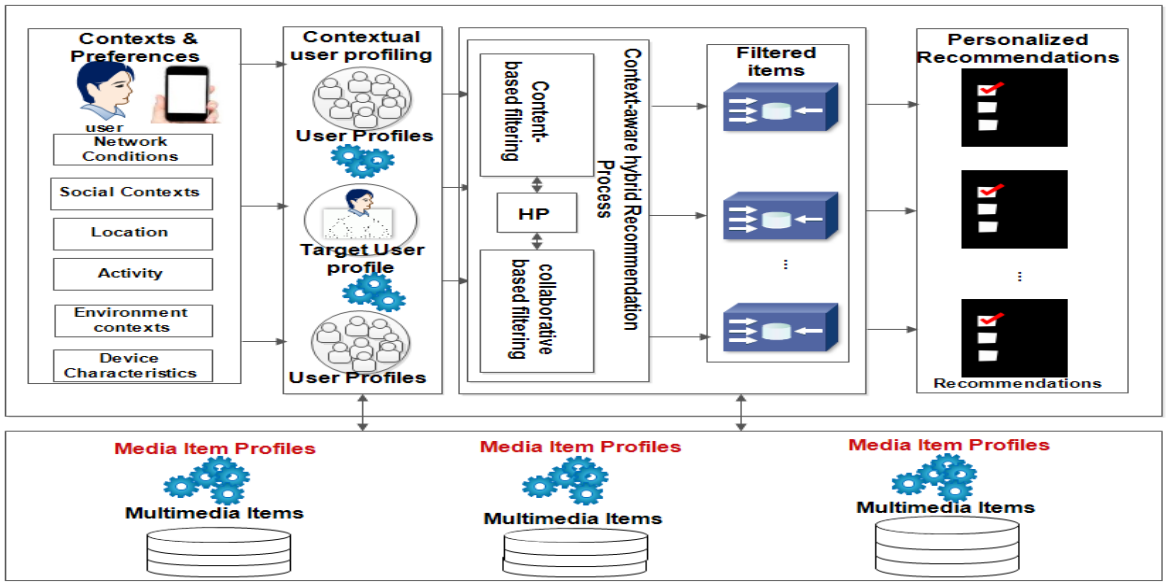


Figure 4.20- CAMR context-aware hybrid process workflow. HP in the figure means hybrid process.

It uses the conventional cosine equation (4.12) to compute the distance between these vectors and ranks them based on the similarity values.

(3) In the third phase, the system generates predicted preferences for each candidate items for the target user, by finding unseen items that are similar to the items obtained in the second step above, using (4.17). Equation (4.17) is a weighted linear of the content based and collaborative based relevance predictions. Thus, a relevance p_{nu_n,i_n,c_n} for every candidate item i_n in context c_n is predicted for user u_n using (4.17), which is a weighted combination of the content based and collaborative-based recommendation predictions of section 4.5.1 and 4.5.2. u_n is the target user for whom recommendation is being computed.

$$p_{nu_n,i_n,c_n} = p_{u_n,i_n,c_n} + p_{ni} \tag{4.17}$$

Where $Pni = \frac{\Sigma_{i=1}^{n} Sim(\vec{v}_c, \vec{v}_m)\, w_{i,m}}{|\Sigma_i^n w_{i,m}|}$

i.e. the prediction obtained from the context-aware content-based process, using equation (4.13) of section 4.5.1.

In addition, $p_{u_n,i_n,c_n} = \overline{p_{i,c}} + m\sum_{u=1}^{k}\left(p_{nu_c,i,c} - \overline{p_{j,c}}\right).\ p_{ni}(u_i,u_j,c \in C)$ is the prediction obtained from the context-aware collaborative process of section 4.5.2.

This hybrid prediction discounts the contribution of every neighbor's preference value (intensity) according to its degree of similarity with the target user contexts so that more neighbors have a large impact on the final preference predictions [38]. The recommendation set is computed by ordering filtered relevant multimedia items in descending order of magnitude of the computed prediction values. Figure 4.21 summarizes the entire hybrid content classification process. However, if no users in the profile repository match the current user on the bases of their contextual information and the context of the current user, non-contextual user profile is computed for the current user. In this case, a complete default user profile is built. The computed user profile is then processed following the traditional recommendation processes to obtain recommendations for the current user.

Process: context-aware hybrid recommendations

1. ***Inputs: target user's current contextual information, contextual user profiles, current user's contextual profile, media item profiles***
2. ***Output: a set of contextual recommendations***
3. ***Sense and classify active user present context***
4. ***Compute the similarity between target user's current contexts and those in the other users' profiles as described in section 4.4.3***

a. ***If context similarity*** $\boldsymbol{f(C_i^l, C_i^n)}$

- i. ***Generate contextual profile for each user***
- ii. ***Generate contextual profile vector for every other user in the profile repository***
- iii. ***Generate N users(neighbors) similar to the target user based on contextual information***
- iv. ***Filter all content consumed by each neighbor in such contexts***
- v. ***Generate a weighted preference for each neighbor based on (iv)***
- vi. ***Retrieve media metadata from the Internet based on the prediction in (v) using the preference keyword***
- vii. ***Generate media content profiles***
- viii. ***Rank the candidate content for each similar user***
- ix. ***Generate a preference prediction for active user for every media item (4.16)***
- x. ***Rank the TopN candidate content from (vii) for the active User***

b. ***If no match is found:***

- i. ***Generate non-contextual user profile vector***
- ii. ***Generate non-contextual user profile vector for every user in the profile repository***
- iii. ***Generate all friends of the active user(using the non-contextual user profile vectors)***

iv ***Repeat a(iv-x) above***

c. ***Present suggested items to the target user***

d. ***record feedback(implicit/explicit)***

e. ***Log the present context and update his user profile***

Figure 4.21- Context-aware content based collaborative process

4.6 Recommendation presentation adaptation service

After recommendations have been generated and an active user has selected at least a relevant item from the set, it is possible that the user's device characteristics or network conditions would constrain successful display of the selected content. In this case, incorporating adaptation service is necessary. Therefore, to tailor the recommended media item, with characteristics that might not be supported by the device properties and network conditions, the thesis proposed the adoption and incorporation of existing media adaptation service. A good number example of standardized multimedia adaptation mechanisms have been developed by researchers in the field of Universal Multimedia Access (UMA) [6],[127],[147]-[148], [153] and can be incorporated into our proposed framework to provide adaptation service to tailor the recommended item's presentation to the prevailing network conditions and to user's device characteristics. Our proposed adaptation service can profit from existing adaptation service such as MULTICAO to provide the adaptation functionality [127]. The recommended media item presentation adaptation service, therefore,

would be responsible for adapting media item content based on summarization and transcoding techniques [153]. The summarization extracts and presents important parts of content such as audio or video into short ones, according to the battery power remaining, network bandwidth functions. Whereas media transcoding would transform the content from one media type to another based on the network conditions such as bandwidth and according to the device capabilities such as screen size or according to the type of media it is capable of playing. A good example is the *Context-Aware Ontologies for Multimedia Applications (*MULTICAO), which provides adaptation services using ontology based adaptation knowledge model.

Barbosa et al. [153] developed MULTICAO, using OWL DL with adequate provision for adaptation decisions, relying on various rule-based reasoning mechanisms that support automatic inference over adaptation decision knowledge base. In addition, MULTICAO provides Adaptation Decision Engine (ADE), which is responsible for adaptation reasoning and decision-making services and a set of Adaptation Engine operation services responsible for executing adaptation of multimedia content, such as modification of the content into different modalities such as text, video, or images depending on the device and network constraints [127]. It provides a context service manager that collects low-level usage characteristics, extracting relevant values and instantiating it in the core ontology, integrating it with its interrelated concepts and inference rules. The adaptation engine uses the information provided by the ontology to infer situations that might require adaptation operation. Those inference rules for deciding and performing adaptation decisions are invoked whenever new or updated contextual constraints are available, interacting with the decision module to select appropriate adaptation based on adaptation parameters provided. Additionally, it provides adaptation engine stacks, which encloses several adaptation engines into a single entity.

4.7 Summary

The limitations of existing realizations of traditional and context-aware recommendations were described in chapter two. One of such limitations is the use of limited and static contextual information in the provisioning of recommendations. Because users have to provide contextual information and preferences explicitly during recommendation process, these systems are deficient in providing effective and relevant media suggestions to mobile users. Additionally, they are not able to address the *coldstart* problem because they rely on user rating information to compute similarity between user contexts in order to infer user's dynamic preferences. In this chapter, we proposed and described context-aware personalized multimedia recommendations in mobile environments incorporating the dynamic context information provided by the context framework for proactive contextual recommendations. The proposed conceptual framework incorporates contextual information to extend traditional recommendation processes via a new contextual user profile model. This chapter also discussed how the proposed framework addresses *coldstart* problem, using contextual information of neighbors. It also described the adoption of adaptation service, which could be incorporated into the framework to tailor the presentation of recommended content according to the user's device's constraints as well as network conditions. The recommendation processes, being similar to a retrieval problem, where the goal is to identify content that is similar to the query, adopted k-Nearest Neighbor (kNN) method.

The query is the contextualized user profile, whereas the candidate universe is composed of publicly available content repositories. There are two main NN search approaches: 1) linear scan algorithms; and 2) reduced-space algorithms.

The former performs a comparison over the entire data set; the latter uses some spatial data structure, such as trees, or a hashing technique, to reduce the universe of candidates, then applying a linear comparison over this reduced data set. Well-known techniques of the reduced space approach are the kd-trees and several variants such as Best-bin-First (BBF), Sphere-Rectangle (SR) Tree or Spill (SP) Tree and the Locality Sensitive Hashing (LSH) algorithm. In this thesis, Vector Space Model and Cosine Similarity as well as Pearson algorithms were adapted and extended with contextual information based on their unquestionable simplicity and accuracy, as well as their ability to perform competitively with other more complex algorithms (e.g. SVD, PCA, etc.) as discussed in [10] [182]. Finally, the operations of the conceptual framework are summarized next.

(1) It initiates a communication with the contextual user profile service (CUPS) to obtain the user's present contextual situation and corresponding preferences. The CUPS then communicates with the context service manager to obtain

the user present contextual situation, which CUPS uses to process the user profile to obtain the user contextual preferences and sends it to the CAMR.

(2) CAMR determines the specific category of media items (e.g. news, movies, music, etc.) and uses this information to invoke the appropriate recommendation service. It implements three contextual recommendation services. Context-aware content based recommendation service, context-aware collaborative recommendation service and context-aware hybrid recommendation service.

(3) If the recommended media item's characteristics match the device and network constraints, then CAMR invokes the media item presentation to deliver the recommended media items to the user.

(4) If the recommended media item characteristics do not match the device and network constraints, and then CAMR invokes the media item adaptation service to adjust the format, modality, etc. of the recommended items to suit the device and network constraints.

In chapter 5, we present the software design and technologies supporting the implementation of the proposed framework and discuss how it can be realized, and in chapter 6, we evaluate the functionality provided by CAMR, through measurements and data obtained when conducting real-world experiments using the developed prototype.

Chapter 5

Design and Implementation

5.1 Introduction

This chapter describes the implementation work that was developed during the course of this thesis. The development of an actual system implementing the processes and algorithms presented in chapters 3 and 4 is deemed vital to demonstrate their feasibility to support context-aware mobile multimedia personalization. For that purpose, a context-aware personalized recommendation system for mobile users was developed based on the concepts of CAMR. Such system was used to conduct experimental evaluation and thus assess the adequacy of the formulated concepts and approach. The results and analysis of each experiment are presented in chapter 6. The contributions of the present chapter are two-fold. First, it presents the design details of the developed software system. Second, it describes the technologies that can be used to implement such system. The development processes employed in the design of CAMR are categorized into four as depicted in Figure 5.1. These are usage requirement analysis, system design, system implementation and evaluation. The first three processes are presented in this chapter, whereas the last is presented in chapter six.

5.2 CAMR design and development processes

Figure 5.1 depicts a simplified development process and phases of the proposed framework for context-aware personalized mobile multimedia consumptions. The phases as illustrated in Figure 5.1 follow a generic software development process. The first phase is the requirement analysis. In order to understand the basic requirements and functionality expected of the framework, this phase of the development paints a real-life situation in which the proposed solution can be applied. Essentially, this will help to clearly identify the actors such as people, environment, and other systems, etc. that interact with the proposed system, and to understand the nature of interactions between them. Additionally, it will help to define a scope for the proposed system, and to ascertain the feasibility of its eventual implementation. Usually, in software engineering, this is done by interviewing the customers or business owners and the prospective users of the proposed system. For example, scenario of users utilizing the proposed application to consume customized and personalized media items in various contextual situations can be painted and then this scenario is used to elicit key functionality and requirements of the system. The knowledge gained in this phase is then used in the next phase, which is the design phase. In the design phase, the focus is on the distinct features of the proposed system, which are abstracted into the system architecture and various models. The design phase translates the product of the requirement analysis phase into concrete representation of the system, which can be assessed and analyzed to realize its functionality.

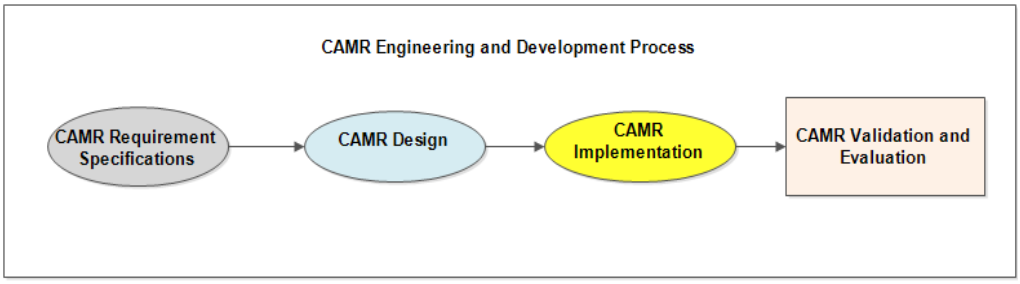


Figure 5.1- CAMR development process

Thus, we have adopted four design processes in the design phase, which are architectural design, scenario and requirement analysis, object oriented design and sequence models. The implementation phase involves the actual translation of the design into classes and from classes into modules or packages, which could then be translated into codes, the process that leads to the working system. In the evaluation stage, actors can then use the product of this phase to perform evaluation of the system.
This last phase deals with the testing, evaluation and validation of the system to ensure that the system meets its design requirements and fulfills its basic functionality.

5.3 Usage scenario

The volume of media items available to mobile users on the Web continues to grow at astronomical rates. Thus, for an average user, finding suitable media items can be very frustrating and time consuming. A recommendation process specially built for mobile environments that seamlessly react to the user's contextual situations can efficiently address this problem. This kind of system is useful for different classes of users, such as students, professors, workers, professionals, etc. Let us take a university student, who performs daily activities ranging from going to the university in the morning to jogging in the evening at the sport complex, consuming various types of content during those activities. More specifically, the following is an archetypical university student scenario, a comprehensive version of the one given in previous chapters. *Ana, a student from the University of Porto, is enrolled in the Electrical Engineering course at the Faculty of Engineering. She lives off campus, at the city outskirts, and catches every morning a train at 7:30 AM that takes her near to the faculty campus in time to attend the 9:00 AM lecture.*

Before leaving home, while taking her breakfast, Ana normally checks the weather forecast. On her way walking to the train station she usually listens to nice country music, whereas on the train, Ana prefers to read morning breaking news. From the train stop, while walking to the faculty campus she normally also profits to listen to some of her favorite music, but when in the classroom she always puts her mobile phone in silent mode. On her way back home in the evening, she also likes to listen to music, except on Fridays when she eagerly seeks for good movies playing in the theaters near her home or in downtown Porto if the weather forecast has indicated dry weekend.

Now let us analyze a specific day in Ana's life, assuming the occurrence of specific contextual conditions:

It is a Friday morning and Ana is taking her breakfast at home. The system on her mobile device automatically pushes information about the weather for the weekend and Ana (as well as the system) realizes that it is going to rain in the morning but dry in the evening. The minute she steps out of her door and starts walking to the train station, the system detects that Ana is walking and that she is using her headphones and consequently it tunes to her favorite Internet radio channel with country music. When she arrives at the train station, she is lucky enough that her train arrives immediately. The system detects again a change in the contextual conditions, informing that Ana is on the train. Accordingly, it searches for news feeds and suggests them to Ana. On her way from the train station to the faculty, the system tunes again to the radio station and switches the mobile phone to the silent mode when she enters the classroom. In the evening after all lectures, Ana decides to go home. On her way home, Ana receives a beep from her phone providing her with latest movies, knowing that it is a Friday evening and that Ana enjoys going to the Cinema on dry weekends. The recommended movies are listed with options for Ana to check the synopsis of each recommended movie in order to select the suitable one for that evening. When Ana arrives at home, sitting in her sofa, the system provides her a more comprehensive set of movies with the actual preview movie clips/trailers. However, while playing the clips, the system realizes that Ana's mobile phone is running out of power and thus it automatically presents Ana with beautiful posters of the movies instead of the clips. Additionally, Ana likes to make long bicycle rides with her friends in the countryside at the city outskirts during sunny weekends (on Saturdays). If the weather forecast announces sunshine for the weekend, then the system provides her with a list of alternative tracks for cycling on Saturday. In addition, when Ana is jogging, the device automatically presents her with latest pop and jazz music, which she likes so much. Finally, Ana also has the habit of jogging with her friend during dry weekdays, especially on Mondays, Tuesdays and Wednesdays.

The application scenario as presented (a detailed version of the scenario presented in chapter one) illustrates a practical example of the application of context-aware personalized media recommendations. By analyzing this scenario, we establish that in order for the system to realize the functionalities in the scenario, we can distinguish various roles that must be fulfilled. Figure 5.2 illustrates an activity model representing various phases and processes of the system realizing the scenario. Each of these processes is discussed fully in the subsequent sections of the chapter. For example, we can identify recommendation service, context service, adaptation service, user profiling service, etc. Thus, the context-aware recommendation system should have the capability to recognize and characterize the contextual situation of Ana, or the context conditions associated to Ana while using her handheld device. In practice, it should be able to identify the kind of activity that Ana performs, such as sitting, walking, jogging driving, etc. as well as her locations, e.g. home, school, train station etc. In addition, of course, it should recognize the time, e.g. of the day, and even the day of the week, such as Friday.

The scenario presented above allows identifying various kinds of contextual information necessary for the system to derive this kind of high-level context information and thus provide contextualized recommendations. Specifically, user identification (who: "userID#"); basic timing data (when: "Friday at 7:30AM, 9:00AM"), allowing to derive more information related with time (evening, weekday, weekend); location information (where: Latitude: 41°10'40.30"N, Longitude: 8°35'54.29"W, etc. could represent the train station, university of Porto, etc.), which contributes to inference of more information related with location (metro station, outside/inside, public space); type of terminal device (terminal: "mobile phone", "tab", "laptop"), device capabilities and conditions(namely the battery level and the type of network connection); movement information such as velocity, acceleration, orientation, etc., which could allow to infer user's activities (e.g. "sitting","walking","jogging","running, etc.), digital item("music", "movies", news", etc.).

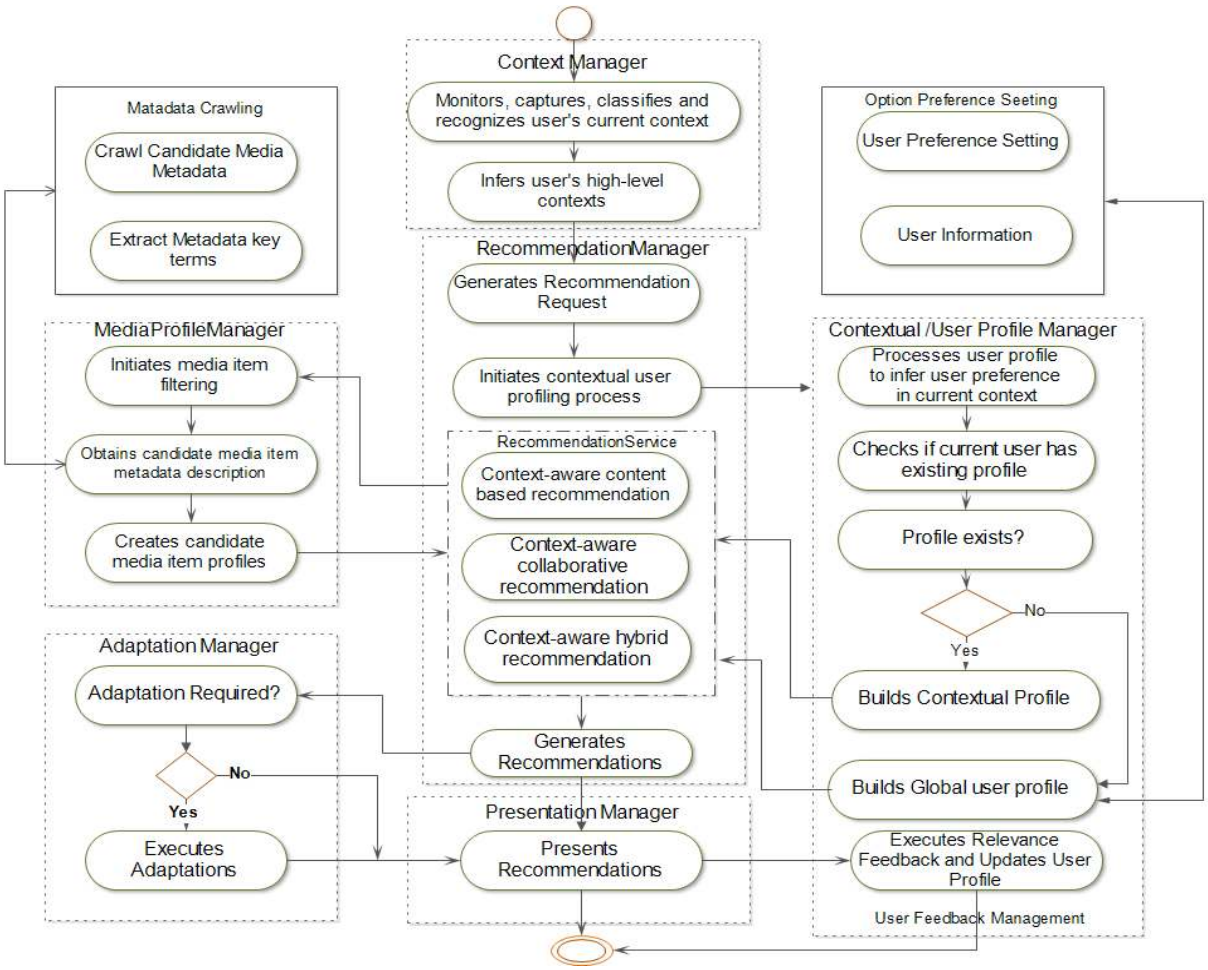


Figure 5.2- Summary of the activities of the proposed system for realizing the usage scenario.

Empowered with this contextual information, together with the preferences of Ana stored in her contextual profile, the system will be able to suggest media items adapted to Ana current context. Thus, to realize all these features, some basic processes are very important. First, dynamic context recognition and inference. Second, contextual user profiling. Third, dynamic and context-aware media recommendations based on the contextual profiling process. Last, but not least, media adaptation.

In the next sections, we use this scenario as a generic use case to derive various functional and non-functional requirements of the system. However, let us first introduce the high-level conceptual model of the system, illustrating its subsystems and main actors.

5.4 CAMR high-level conceptual design

The primary goal of CAMR framework is to support both implicit and explicit provisioning of adaptive personalized context-aware mobile multimedia content recommendations for mobile users. In order to realize this goal, CAMR provides an infrastructural support for context-aware personalized recommendation services that deliver relevant multimedia content to mobile users. CAMR provides two main subsystems, as illustrated in Figure 5.3, from high-level view, the CAMR client and CAMR server subsystems. The CAMR client subsystem provides on behalf of mobile users, on one hand, the initiation of explicit content recommendations. It does this in situations where users make request for relevant multimedia content by collecting, for example, certain keywords from the user and then determines the user's contextual information, the user's basic preference information, sending this information to the CAMR subsystem.

The CAMR server subsystem then processes the request and returns a list of relevant multimedia content to the user via the CAMR client subsystem. On the other hand, it can initiate the personalized recommendation process based on the user's contextual changes without the user's involvement. A service provider (another actor) can subscribe to services provided by CAMR subsystems or integrates CAMR subsystems as part of the services they provide for their service consumers. In this case, their service consumers must be provided with CAMR client subsystem, installed on their devices such as mobile phones, Tablets, or and other handheld devices. Thus, the high-level conceptual architecture of CAMR subsystems consists of four main actors as illustrated in Figure 5.3: CAMR service client/CAMR user, content provider, service provider and CAMR administrator. These actors have been identified at higher abstract level as people or things that interact with CAMR system and to whom it delivers values. These actors are the things outside the system that interact with it to deliver values it provides. The CAMR service client in most cases act on the CAMR system on behalf of the CAMR user.

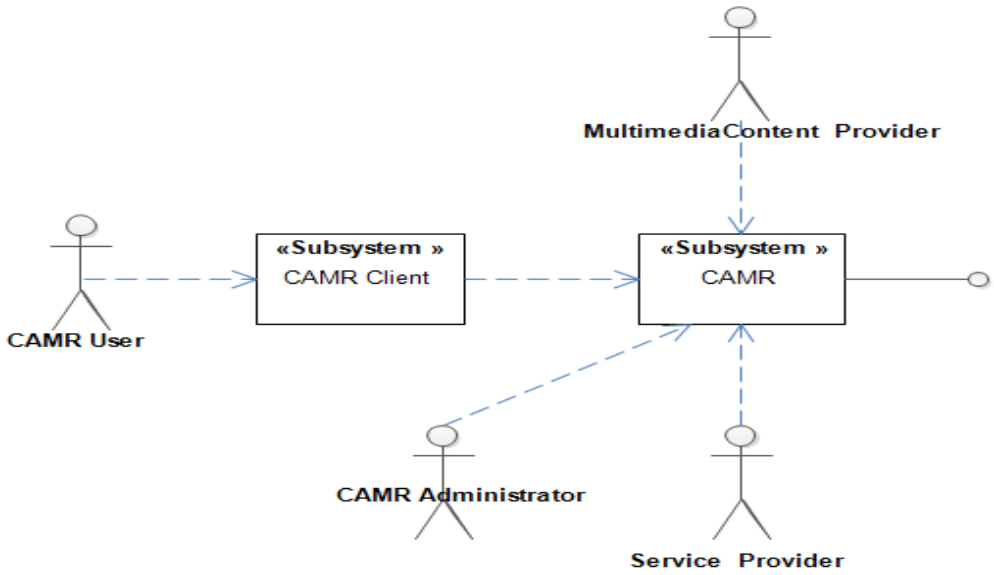


Figure 5.3- High-level conceptual architecture of CAMR showing its primary actors and subsystems

5.4.1 CAMR user and CAMR client service

The CAMR Service client represents one of the two subsystems making up the CAMR system. The CAMR user, a human actor or a third party application, uses the CAMR Service client to consume services provided by the CAMR platform. The CAMR user may or may not explicitly send request to the CAMR system to benefit from the values it provides such as personalized recommendations of multimedia content. Through the CAMR service client, the CAMR user initiates either recommendation process or CAMR service client implicitly delivers personalized recommendations to the CAMR (mobile) user according to his contextual changes. The CAMR Service client interacts with the CAMR subsystem exchanging information such as:

- *Contextual information, user credentials such as **id**, optional user information such as **gender**, basic preferences, etc.*
- *Request for multimedia content recommendation or search*
- *A set of recommended multimedia items*
- *A request to download and play user selected recommended media items*
- *User feedback information*

5.4.2 The content provider

This third party system owns the rights of the multimedia content and makes multimedia content available online. They are also responsible for the creation, storage and management of multimedia content. Multimedia content service providers can implement basic interfaces provided by CAMR or CAMR can access multimedia content via interfaces provided by the multimedia content providers or through service provider (described next). The interactions between content provider, service provider and CAMR service client include but not limited to the:

- Provisioning of online accessible interface to candidate or consumed multimedia content
- Registration of CAMR as multimedia content consumers
- Handling of request by CAMR to obtain multimedia content metadata
- Handling of request by CAMR service client to download recommended multimedia content

5.4.3 Service Provider

Like the multimedia content provider, service provider is a third party system, which is not directly responsible for the creation, storage or management of multimedia content. It only provides access to multimedia content provided by the content provider by implementing interfaces that allow access to different multimedia content it aggregates from many multimedia content providers. For example, online services such as theimdb.com and imdb.com provide access to movie metadata but they are not responsible for movie content creation and management. Thus, the interaction between CAMR and service provider are the same as those of the multimedia content provider as listed above. Additionally, a service provider can also implement services provided by CAMR platform to deliver values to their service consumers.

5.4.4 CAMR Administrator

The administrator represents the CAMR system administrator, performing housekeeping functions, such as knowledge base maintenance, optional user registration, database maintenance, etc.

5.5 CAMR Functional specification

In this section, we present the use case model, functional requirements and specifications of CAMR system platform. The use cases presented are those that are more relevant to the definition of the architecture of CAMR system. Some of the use cases extend or include other use cases. The use case model comprises the actors, that is, those who interact with CAMR and how they will interact with it and the values it delivers to them. The reader should note that in the

use case model as illustrated in Figure 5.4, two actors are provided as opposed to the number of actors shown in Figure 5.3. There are four actors as defined in Table 5.1 where each actor relationship with the developed use cases is defined. The other two actors, the content provider and the service provider are managed by CAMR administrator and therefore have been excluded from the use case model.

5.5.1 System use case model

Ivar Jacobson invented Use Case Diagram, which has become an important part of Object modeling Group's Unified modeling Language (UML) that is now very popular in system modeling. It is used generally to outline the business logic of software development projects [184]. It specifies a set of actions, otherwise known as functions, performed by a system, which yield observable results that are typically of value to one or more actors or other stakeholders of the system. Additionally, it can be seen as a way to specify how users interact with the system, keeping in mind the value such interactions provide to users. It also helps, from the users' perspectives, the value that the system can deliver by helping to distill the functional requirements of the proposed system. In this section, we present the design of a use case model of the above scenario presenting components of the system as use cases with four actors as previously illustrated in Figure 5.3.

Table 5.1- CAMR main use cases , descriptions and actors

Use Case	Description	Actor
1. Get Contextual Information	This use case comprises two processes. First, determined by the user, this use case delivers monitoring, acquisition, and classification of mobile user contextual information. Second, it provides high-level contextual information from basic contexts using semantic model such as ontology and SWRL.	CAMR Service Client
2. Generate Contextual User profile	This use case uses the current context of the user to process the user profile information in order to predict the preference of the user. This use case includes others use cases such as Predict User Preference, Infer Contextual Information. It extends Get User Profile use case.	CAMR Service Client
3. Get Contextual Recommendations	Two types of recommendation are provided by this use case. First, based on the user's direct request by typing some keywords, initiates the recommendation process. Second, based on user preference setting, this use case uses contextual changes of the user to initiate recommendation process without user direct involvement. The primary use case is the Get Recommendations.	CAMR Service Client
4. Present Recommendations	This use case involves the delivery and presentation of recommended items to the users.	CAMR Service Client
5. Get User Feedback	In this use case, first, user can be asked to provide evaluation of the relevance of the recommended multimedia content. Second, user's multimedia content consumption is monitored and his acceptance or not of the provided recommendation is communicated to the server	CAMR Service Client
6. Manage multimedia metadata	The service provider or content provider is requested to provide the metadata of the candidate multimedia content	Service Provider, Content Provider
7. Manage knowledge base	In this use case, knowledge bases such as ontology context model, is managed	CAMR Administrator
8. Manage Users	Users are managed by CAMR Administrator	CAMR Administrator
9. Manage Content/Service Providers	It adds new content or service provider interface, update or delete old ones	CAMR Administrator

We have included only use cases that are more important from the architectural point of view. The architecture of the system builds upon these use cases, adding more functionality to the model. In addition, in the use case model illustrated in Figure 5.4, we have excluded the CAMR Service client actor because it acts on behalf of the user, thus interaction by the CAMR user actor is via the CAMR service client. Figure 5.4 gives an overview of the functionality of the context-aware media recommendation system, the brief description of each of the most important use cases is given in use cases 1 to 9.

These use cases are used to abstract the functionality of the system and for the design of the functional architecture of the system, where each use case is mapped into the function rendered by the component of the architecture. In most of the use cases, the user is not directly involved as an actor; instead, it is the CAMR client on behalf of the user. However, we assumed that CAMR's user is the primary actor because his/her action, via CAMR client running on his/her device, triggers context sensing and recognition. Also, note that the use cases described in Table 5.1 incorporate special cases through inclusion and extension. These inclusions and extensions are also illustrated in Figure 5.4. For example, *Get Contextual Information* use case includes *Infer Contextual Information* use case. Another example is *Get User Feedback* use case, which is extended by two other use cases, *Get Implicit Feedback* and *Get Explicit Feedback* use cases. The observation of a significant change of the user's context as sensed by his device serves as the activator of the contextual recommendation process. This remote action is responsible for the chain of actions, which are consequently fired in the contextual recommendation process.

5.5.2 Use case model specifications and event flow

In this section, the use cases in Figure 5.4 are described using event flow tables. As can be seen in this figure, some of the use cases may not have direct interactions with the actors. However, those use cases that have direct interactions with the actors interact with the former to execute their functionality. In the model, those use cases are specified as either extended or included use cases.

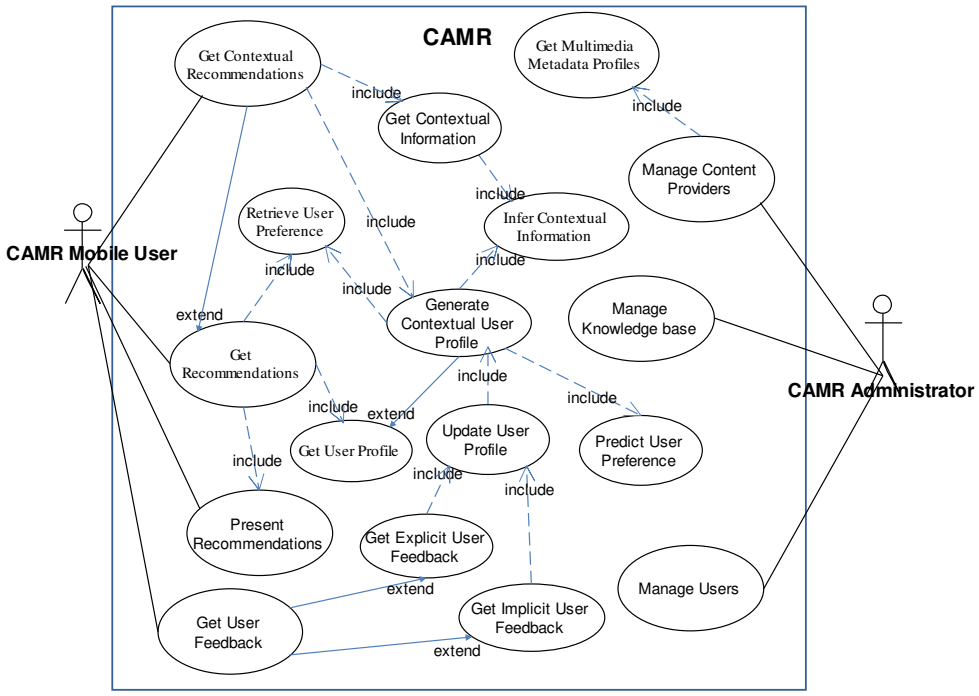


Figure 5.4- CAMR use case model showing the abstraction of the operational level and functional requirements of the proposed framework

(1) ***Get contextual information***

This use case, whose flow of events is illustrated in Table 5.2, describes the monitoring, acquisition and classification of context information. It includes Infer high-level contextual information (see Table 5.3).

The difference between the two is that the infer high-level contextual information can combine one more contextual information through some additional operations to obtain more semantically meaningful contextual information using ontology and SWRL models described in chapter 3.

(2) ***Generate contextual user profile***

This use case describes as illustrated in Table 5.4, the contetual user profile process, which associates user preferences to contexts in which such user preferences were predicted. This use case also includes the prediction of user's unknown preference in the current context i.e. predict user preference use case as well as **Retrieve User Preference** use case. In addition, ***Retrieve User Preference*** This use case describes the retrieval of basic preferences stored on the user device. It should be noted, however, that the CAMR Service client on behalf of the mobile user retrieves this information and sends it to the CAMR server if the user indicates so. This is necessary to allow users to device whether they want their sensitive information such as age, sex, location etc. to be used or not in the personalization process. Table 5.5 illustrates its flow of events. After retrieving user's basic preference, the prediction of user's preference in the current context is launched, Table 5.6 illustrates the flow of events for this use case.

Table 5.2 - Get contextual information use case

Use Case 1	**Get contextual information**
Actor	Mobile User/CAMR Service Client
Description	In this use case, low-level contextual information such as environment context, e.g. illumination, noise level, etc., as well as user activity, location, time of day, day of week, etc. is sensed and recognized by user's device sensors. User is not involved directly or explicitly in this process, but his device is used to obtain the information. Also, many other processes are involved in this use case, which are provided in the event flow.
Trigger	Predetermined interval can be set to report changes in sensed events. It can also set it based on sensing that the user is holding or using the device, which itself is achieved via device sensors such as accelerometer or orientation sensor. It can also be triggered if user explicitly makes recommendation request.
Preconditions	User engaging the device or when predetermined time interval elapses
Postconditions	The information obtained is sent to the server subsystem via the communication interface to the recommendation management service which then starts the recommendation process.
Event flow	1. Context management service activates the device embedded sensors, 2. Context monitor collects sensor data 3. Context aggregator then retrieves data from various context monitors and pre-process them by removing outliers, it then extracts features from data 4. The features are served as input to the context classifier, which then infers the high-level contextual information 5. The classified context information is sent to the server provided user agrees that such information is sent in the initial setting information obtained from such user.
Inclusion	Use Case 3: Infer High-level contextual information
Extension	N/A

Table 5.3 -Infer high-level contextual information use case

Use Case 1.1	**Infer high-level contextual information**
Actor	Mobile User/CAMR Service Client
Description	In this use case, user's contextual information such as natural environment conditions described by his environment contexts, like illumination, noise level etc., as well as other low-level sensory data such as velocity, acceleration, orientation, etc., location, time of day, day of week, etc. are received by the context manager and then instantiated into the context inference model, which then returns a high-level form of the context information as a contextual situation of the user. User does not have to be explicitly involved in this use case, since the process is automated on his device to obtain the information.
Trigger	Request from recommendation management service
Preconditions	Change in user context is recognized and explicit request from the recommendation management service to the context management to obtain high level context
Postconditions	The high context information obtained is sent to the user profiling management.
Event flow	1. A request for high-level context inference by the get contextual recommendation or get recommendations is received 2. Instantiates received contextual information into the inference mechanism 3. Performs inference 4. Returns inferred contextual information to the get contextual recommendation management service
Inclusion	N/A
Extension	N/A

Table 5.4 - Generate contextual user profile use case

Use Case 2	**Generate contextual user profile**
Actor	Mobile User/CAMR Service Client
Description	In this use case, the user profiling process begins with the determination of the user's preference. The contextual user profile takes preference and the contextual information as inputs and builds a contextual user profile, which then is filtered to produce a contextual user profile vector used subsequently in the contextual recommendation process.
Trigger	Get Contextual recommendations, get recommendations
Preconditions	Retrieve user preference
Postconditions	Contextual user profile/Contextual user profile vector
Event flow	1. The contextual user preference is used to build a contextual user profile 2. The contextual user profile is filtered in ordered to obtain contextual user profile vector 3. Contextual user profile vector is returned to the recommendation process.
Inclusion	Predict User Preference, Retrieve User Preference
Extension	Build complete user profile without contextual information use case. This extension occurs if contextual user preference was not returned by the infer user preference use case. In other words, no contextual information is available.

Table 5.5 - Retrieve user preference use case

Use Case 2.1	**Retrieve user preference**
Actor	Mobile User/CAMR Service Client
Description	Although it is not mandatory, this use case allows users to specify their basic information if they so wish. Optional basic setting information, such as age, sex, profession, etc., is obtained. Additionally, users can set frequency of context monitoring, some other basic static contexts, e.g. point of interests.
Trigger	Get Recommendations/Get Contextual Recommendations
Preconditions	Context changes or user request for recommendations
Postconditions	The information obtained is stored in the user profile repository
Event flow	1. If user chooses to provide the basic user information, the system uses this information to create user's initial contextual user profile 2. If users choose not to provide this information but still want to subscribe to the service, user's current context can be used with existing users who have been in the same context to create initial contextual profile
Inclusion	N/A
Extension	N/A

Table 5.6 - Predict user preference

Use Case 2.2	**Predict contextual user preference**
Actor	Mobile User/CAMR Service Client
Description	This use case obtains user's contextual preference, which it then uses to build and filter contextual user profile via the contextual user profile management, if the user's contextual information is available. Otherwise, complete user profile and context-free user preference is applied.
Trigger	It triggers when recommendation request is made followed by infer high-level contextual information request, from the user case 4
Preconditions	Recommendation Request, Infer High-level contextual information
Postconditions	The user preference obtained is used to build and filter user profiles. So, the result of this use case is the contextual user preference
Event flow	1. Receives a call from the Get Recommendation use case via build contextual user profile use case 2. Creates and instantiates the user preference reasoner 3. User preference reasoner starts user preference inference 4. Inferred user preference is returned to the process and build user contextual user profile use case
Inclusion	N/A
Extension	N/A

Table 5.7 - Get contextual recommendation use case

Use Case 3	**Get Contextual Recommendations**
Actor	Mobile User/CAMR Service Client
Description	This is a controller use case that initiates and manages the context-aware recommendation processes after it has been observed that user context such as activity or location has changed. Additionally, the user can also explicitly initiate recommendation process, after which user's contextual information is determined before the next use case process is fired.
Trigger	Get Contextual information or Get Recommendation use cases
Preconditions	User high-level context inferred and obtained from the user device, or user explicitly request recommendation.
Postconditions	Though this use case is the core of the entire process, which initiates other use cases, but in the end, a set of recommendation is provided and presented to the user.
Event flow	1. After receiving contextual information from the client, or after user explicitly requests for recommendation, the recommendation process starts a thread. 2. First, it sends a message including the contextual information to the contextual profile management service. 3. It receives a response from the context management service with a higher level contextual information 4. It then sends a message to the context profile management services to infer the user's preference in that context 5. After receiving the preference and contextual user profile from the last process, it then proceeds to start recommendation process 6. Recommendation list is provided and presented to the user
Inclusion	Get contextual information, Infer high-level context information, Manage media profiles, Retrieve user preference, Infer user preference, Generate contextual user profile ,
Extension	Get Recommendations

(3) ***Get contextual recommendations***

This use case describes the contextual recommendation process: Its main function, as illustrated in Table 5.7, is to obtain recommendations for the user based on her current contextual information. The use case is triggered when user's contextual information such as her location, time and activity changes.

It extends the ***Get Recommendations*** use case (see Table 5.8), which is an explicit recommendation process whereby user directly requests for recommendation by typing some keywords. In either case, recommendation can be generated using contextual information (if available) or switch to traditional recommendation mode if contextual information cannot be obtained for whatever reason.

Table 5.8 -Get Recommendations use case

Use Case 3.1	**Get recommendations**
Actor	Mobile User/CAMR Service Client
Description	This use case processes the generation of a list of media items suitable for the user's current contextual situation. First, it generates recommendations based on user's explicit request.
Trigger	Request from start recommendation management service use case.
Preconditions	A list of contextual recommendations
Postconditions	Present Recommendations(list of recommendations)
Event flow	1. User explicitly requests for recommendations by typing some keywords 2. Using the keywords and contextual information, it determines the user preference, and category of the candidate media items, determines the appropriate recommendation process to fire. 3. It provides the recommendation process with a similarity matrix consisting of the user contextual profile vector and media item vectors 4. The similarity between the user profile vector and each media vector is computed 5. The similarity is then used to compute the relevance or predicted preference value od the user for the items 6. The first n items with the highest relevance values are then ranked in a list 7. The list can then be presented to the user as recommendations
Inclusion	Present Recommendation, Infer User Preference
Extension	Get contextual information

(4) ***Present recommendations***

In this use case, recommended multimedia content is presented to the user (see Table 5.9). The user can then select one or more items from the presented set of recommended items. In most cases, this use case also determines how to display or play the selected multimedia content. It provides adequate information explaining the recommended content. Additionally, it includes the explicit user feedback use case.

Table 5.9 - Present recommendations use case

Use Case 4	**Present Recommendations**
Actor	Mobile User/CAMR Service Client
Description	This use case is responsible for the presentation of recommended media items. The presentation can be in text, video, audio or in any other standard formats or their combination.
Trigger	Request from Generate media recommendation use case
Preconditions	Generate media recommendation
Postconditions	Determination of the best format to present the recommended item
Event flow	1. The links to the location of the recommended items are obtained 2. The appropriate format to present the recommendation is prepared 3. Compare the capability of the user device with the formats of the recommended items 4. If the formats match the characteristics of the device: the recommendation is sent to the device to be played 5. if not: the adaptation service is fired
Inclusion	Get user feedback
Extension	N/A

(5) Get user feedback

This use case describes the process of updating the user's contextual profiles (illustrated in Table 5.10). It extends *Get Explicit* and *Get Implicit* user feedback use cases. It is triggered when the user has consumed multimedia content. Another use case included in the Get User feedback use case is the ***Update user profile*** use case illustrated in Table 5.11.

Table 5.10 - Get User Feedback use case

Use Case 5	**Get user feedback**
Actor	Mobile User/CAMR Service Client
Description	In this use case, user's interaction with the recommended media items is tracked via a relevance feedback mechanism., e.g. using negative or positive response of the user to the provided recommendation via an explicit feedback mechanism, or a continuous monitoring of user consumption via an implicit mechanism. The obtained information is used to update the contextual user profile.
Trigger	Present recommendations, consumption or the delivery of the recommended media items by the user
Preconditions	Get Recommendations, Get contextual recommendations, Present Recommendations
Postconditions	User's latest preferences are ready to be incorporated into the contextual user profile
Event flow	1. After the presentation of the recommended item to the user, user consumption or not of the recommended item is tracked either implicitly or explicitly 2. If user is willing to provide rating of the recommendations, 1. explicit feedback process is fired 3. Else: 1. Implicit user feedback is fired. 4. the feedback process sends the result to update user profile update
Inclusion	N/A
Extension	Get Implicit User Feedback, Get Explicit User Feedback

(6) *Manage multimedia metadata*

This use case describes the process that obtains candidate or consumed multimedia content metadata from external sources, which includes content providers and/service provider. In addition, this use case also processes the metadata and produces profile for each candidate media content in readiness for use in the recommendation process. Table 5.12 illustrates the flow of events of this use case.

Table 5.11 - Update user profile use case

Use Case 5.1	**Update User Profile**
Actor	Mobile User/CAMR Service Client
Description	In this use case, after user has received the recommended items, the system begins the process of contextual user profile update via "user perform relevance feedback" use case. The update user profile use case fires the get user feedback use case by sending information about the consumed media item. This is essentially a process that learns user consumption preferences over time and uses this knowledge to improve on the quality of future recommendations.
Trigger	Consumption of recommended media items or delivery of recommendations
Preconditions	Present recommendation
Postconditions	User profile is updated
Event flow	1. Fires the relevance feedback to collect relevant consumption information from the user either explicitly or implicitly. 2. If user provides directly feedback when prompted by the system, explicit user feedback process is executed 3. Otherwise, implicit user feedback is executed. 4. The relevance feedback result and the context of consumption are then used to update the user profile to reflect up-to-date contextual user profile information.
Inclusion	Get user feedback
Extension	N/A

Table 5.12 - Manage multimedia metadata use case

Use Case 6	**Manage candidate multimedia content metadata**
Actor	Mobile User/CAMR Service Client , Service Provider, Content Provider
Description	In this use case, metadata of candidate media items are collected from external media content providers from the online or internet based servers. The collected metadata maybe in different description formats such as XML, or MPEG-7 MDS or MPEG-21-DID.
Trigger	Request from the start recommendation management service use case
Preconditions	Generate contextual user profile use case
Postconditions	Media item vector which represents the candidate media items
Event flow	1. After receiving a request from the recommendation management service, the filter media profile use case is fired. 2. A request is sent to the request media metadata use case is sent 3. Media metadata use case initiates and contact external content providers to crawl and parse the candidate media metadata 4. The filtering process of the obtained metadata is fired. 5. Filtered media metadata/media vectors are returned
Inclusion	Get media metadata profile use case
Extension	N/A

(7) ***Manage knowledge base***

This use case describes the maintenance of the knowledge base (Table 5.13), which involves addition, deletion or updating of concepts in the knowledge base, especially those involving ontological concepts, rules etc., which are used to infer high-level contextual information.

Table 5.13 - Manage knowledgebase use case

Use Case 7	**Manage Knowledge base**
Actor	Administrator
Description	In this use case, information such as context concepts etc. in the knowledge base is reviewed and updated by the administrator. For example, the ontology concepts, rules etc. are constantly reviewed and updated
Trigger	N/A
Preconditions	N/A
Postconditions	The information obtained is instantiated into the ontology model for inference process.
Inclusion	N/A
Extension	N/A

(8) ***Manage Users***

This use case manage describes the management of user information by the administrator of CAMR (Table 5.14). Although users have the option of either sending or not their sensitive information, but important user information needed for recommendation processes such as context history, preferences. It also ensure safety and trustworthiness of users.

Table 5.14 - Manage user use case

Use Case 8	**Manage User**
Actor	Administrator
Description	In this use case, information such as user profile information etc. in the knowledge base is reviewed and updated by the administrator. For example, creation of user information and ensuring that this information is not compromised.
Trigger	N/A
Preconditions	N/A
Postconditions	The user information is updated in the database.
Inclusion	N/A
Extension	N/A

(9) ***Manage Content Provider/Service Provider***

This use case describes the information involving inclusion or deletion of multimedia content providers, illustrated in Table 5.15. The primary actor is the CAMR administrator who oversees which content providers to include as the CAMR content sources. For example, multimedia content providers may consist of a set of publicly available web service interfaces through which multimedia content metadata can be obtained or through which multimedia content can be downloaded. It adds new content provider or service interfaces, updates or deletes old ones.

Table 5.15 - Manage content provider/service provider use cases

Use Case 9	**Manage Content Provider/service provider**
Actor	CAMR Administrator
Description	It adds new content provider interface, update or delete old ones
Trigger	N/A
Preconditions	N/A
Postconditions	CAMR administrator decides on when and how to add content and service provider information to the CAMR system.
Inclusion	N/A
Extension	N/A

5.6 Design and implementation of context-aware media recommendation framework

In this section, the chapter discusses the design of each component for realizing the above-enumerated services of the CAMR functional architecture, guided by the elicitation of a set of functional and non-functional requirements derived from the application usage scenario. Table 5.16 provides a bird's eye view and descriptions of CAMR requirements and their related use cases. These requirements are discussed in the next sections.

5.6.1 Requirements analysis and design decisions

The design of CAMR software system is governed by a set of requirements. In software engineering, requirements can be classified into two categories: Functional and non-functional requirements [184]. On one hand, functional requirements of a system specify what the system or its components are expected to accomplish in order to realize the services it provides. These requirements, such as context sensing, context inference, context-aware user profiling, context-aware media recommendations, and recommendation adaptations in CAMR, as well as other processes that are involved in the overall functioning of the system have to be specified. On the other hand, non-functional requirements, rather than focusing on the functional behavior of the system (what the system does), focus on how the system performs its functions.

They can be seen as quality features of the system's behavior. Examples of such nonfunctional requirements of the proposed system are its ability to allow users some level of control when using the system. The system should be able to collect and use contextual information to predict user's preferences, such as Ana enjoying pop music while walking, without asking Ana for the activity she is performing.

Other examples of non-functional requirements are trustworthiness and the ability of the system to ensure that the privacy of users is not compromised. Another non-functional requirement is the ability of the system to provide recommendations in reasonable time, i.e. minimal recommendation time. Additionally, a software supporting generic context-aware personalized recommendations, like CAMR, should be able to integrate different recommendation techniques, with the appropriate level of abstraction. Now, let us elaborate on these functional and non-functional requirements of the proposed system based on the analysis of the provided usage scenario above.

Table 5.16 - List of requirements and related use cases

Requirements	Description	Use Case
Context sensing and recognition	CAMR must be able to sense, acquire, classify and infer user's contextual information	Get contextual information
Communication of contextual information from CAMR client service to CAMR Server	Contextual information should be communicated to the CAMR system for contextual user profiling, contextual recommendations and user feedback	Get contextual information
Transparent and dynamic user's contextual preference	Processing of user preferences based on the user's contextual information.	Generate Contextual user profile
Contextual recommendations	Contextual recommendations suggest multimedia content to users based on context in which they preferred content in the past or based on the context of like-minded users. Also, it deals with user explicit recommendation by searching for multimedia content	Get Contextual Recommendations
Communication with external resources	Communication with service and content providers deals with collecting multimedia content metadata as well as downloading and playing recommended multimedia content on the user's device	Manage content providers and Manage service provider
Recommendation presentation	This requirement deals with playing of recommended multimedia content in format suitable to the user's device.	Get Contextual Recommendations
Explicit and implicit user profile update	User feedback collects the appraisal of recommended content by the user. It supports implicit feedback in which users do not have to intervene in the feedback process. It equally supports explicit feedback where users are asked to provide feedback directly after consuming the recommended content	Get User Feedback
New user issue	This requirements addresses the problem of new user by identifying the contexts of the new user and then using contexts of existing users similar to that context to provide recommendations	Manage Users, Get Contextual Information

5.6.1.1 Functional requirements

Functional requirements describe functions the system is expected to perform. Based on the development principles of the conceptual framework presented in chapter 4 and the usage scenario presented in section 5.1.1, the following functional requirements have been identified.

1. *Context sensing and recognition*: The system supports incorporation of context sensing mechanisms as well as context reasoning and inference mechanisms as first-class functionality to bridge the critical gap between context-awareness and recommendation processes. The knowledge of user's situations such as his movement velocity, acceleration, location, environment conditions (such as illumination and noise level), device characteristics, and network conditions should be obtained and represented in a standardized machine processable format that can be communicated between various components of the system, either distributed or not. Additionally, the system should be able to infer from these atomic contextual features, more meaningful or high-level contextual information that can be utilized by other components of the system such as contextual user profile model or the recommendation processes. For example, in the scenario given above, the framework should be able to detect user's activities, such as walking, sitting, etc., as well as the user's location dynamically. For instance, when she is at home, train station or in the university, this information should be obtained without asking for it from the user. This information could be used to infer the user's consumption preferences.

2. *Communication of contextual information from CAMR Service Client to CAMR*: The context information as stated above has to be presented in machine processable format e.g. in standardized XML based descriptions such as MPEG-21 UEDs and MPEG-7 media descriptors as described in chapter 3. This is important because we assume that various components of the system are distributed, possibly running on disparate hardware and software, as well as network platforms. Thus, this information could be shared between heterogeneous components, requiring standard protocols and common representation. Existing recommendation systems do not represent context information in a

way that allows interoperable communication of context information between its components. Standardized representation of contextual information for communication between sources of contexts and recommendation systems is an important requirement.

3. ***Transparent and dynamic user's contextual preference***: The major weakness of existing context-aware recommendation systems is that of explicit specification of user preferences and contextual information that requires frequent user interventions [179]. However, a truly context-aware recommendation system requires minimal user interventions in order to ensure user's confidence and trust. User preferences should be managed in such a way to learn them from the user's current context or previous contexts (if available) including contexts of those users who are similar to the current user.

4. ***Contextual recommendations***: Based on the newly inferred user's high-level contexts, suitable recommendations are provided to the user. Thus, the application should determine media items that best match user's preferences in the present contexts. Additionally, it should be able to combine recommendation processes such as context-base content recommendation and context-based collaborative based filtering into a hybrid recommendation process if the need arises. This is necessary to mitigate the weaknesses of the traditional recommendation processes. For example, assuming that Ana in our scenario is a first time user, who does not have adequate profile information in the system, in this case, context-based collaborative recommendation can be used by using information about users who have consumed content in the context similar to Ana's present context, including content of items they consumed in those contexts.

5. ***Recommendation presentation****:* The system provides users with an interface where recommended media items can be accessed with adequate information on each recommended item. For example, media items such as movies can be provided in various presentation forms, such as in clips, audio, synopsis or even using posters.

6. ***Explicit and implicit user profile update***: The system provides users with a medium to assess the recommendations they receive via a feedback mechanism. This mechanism should allow flexibility in the way such information is obtained from the user. Thus, the explicit user feedback of CAMR allows users the ability to express directly how they feel about the services provided by the system. On the other hand, since users might not always want to give this kind of feedback every time recommendation is provided to them, the implicit feedback mechanism is included to monitor the user behaviors when receiving recommendations. In this case, the system monitors user's interactions with the system via a background monitoring service, which reports on the usage of the system by the users. It should be able to remember contextual situations in which the monitored user behaviors occur so that this information can be used to predict future preference of the user. In CAMR, the implicit and explicit feedback processes were discussed in chapter four. For instance, it provides a model that monitors the frequency of consumption of media items with certain key terms, genre, or category, including the corresponding context of consumption.

7. ***New user issue***: The system should be able to provide recommendations even when ratings and feedbacks are not available. In situations where there are no ratings, existing solutions have found it difficult to provide satisfactory recommendations. In CAMR for example, contextual data of the likeminded users are used to address the new user problem as described in chapter four It uses contextual situations (which are similar to the current user's context) in which existing users have consumed items in the past using a context-aware collaborative or hybrid recommendation process.

8. ***Communication with external resources****:* The system also supports the integration of resources from diverse external sources, such as the Web, via published APIs and Web services. For example, the system is designed to provide recommendations of media items, which are not locally available on the system. Therefore, the ability to connect to external content sources and obtain information such as metadata of candidate media items is an important functional requirement. The obtained metadata information is used to develop media item profiles, which are utilized in the recommendation processes.

9. *Interoperability*: Finally, it is important that the system provides the ability to enable users with diverse kinds of heterogeneous devices, with multimedia and network enabled functionality to consume diverse media items according to contextual situations satisfying their preferences. This could be realized via a Web Service interface, with interoperable platform allowing universal accessibility to resources and services provided by the proposed framework, as well as those provided by external sources.

5.6.1.2 Non-functional requirements

As presented in chapter four, some of the primary functions of the systems are sensing user dynamic contexts, using the device's embedded sensors. These functions impose some constraints or quality attributes on the system's functionality, which should be managed in the design in order to mitigate their negative impacts.

1. *Optimizing power consumption*: Because using device-embedded sensors to acquire user's dynamic contexts involves sometimes continuous probing of sensors, power consumption becomes an important consideration. Some of these sensors, such as GPS or even Wi-Fi consume a lot of power. Mobile devices run on battery, and the battery lifetime is limited, therefore users become sensitive to application that quickly drains their device battery. To limit the power consumption of such context-aware application, the developers of context-aware media recommendations based on handheld device sensing should consider some optimization to minimize, as much as possible, the power consumption requirement of the application. For example, location information can be collected at intervals of time since the location of a user may not change for a specific period of time. Since WiFi does not consume as much battery power as GPS, whenever WiFi is available, the location information source can be collected from the WiFi.

2. *Privacy and trustworthiness:* It is important to consider issues of privacy, especially when the system has to collect information with limited user's interventions. Users are understandably sensitive to tracking and usage of their personal information, such as location or activity. How can users be assured that their privacy and safety are guaranteed? In the design of CAMR, it is ensured that the user enjoys some level of control on how and what the system can access in order to provide them with contextual recommendations. For example, sensitive user information such as user's gender, names, age, and even location information, etc., is kept on the device. The use of names is not compulsory to recommend items to users. CAMR only needs an Id to identify a given user. In addition, only context history can be stored on the server for future personalization. The use of the current user location requires permission from the user. Thus, the user must decide whether to allow the system to use this information. Essentially, this means that if the system uses it, then the user must have granted the permission. Another way CAMR achieved this is by using anonymity in the communication between the client and the server. The server does not need to know the users by their names in order to provide them with recommendations.

3. *Scalability*: Does the system perform well in terms of response time and the amount of data it can process? Because modern devices now come with a level of sophistication that competes with those of traditional desktop systems, some of the processing that traditionally would have been done on the server can now be performed on the device. For example, the proposed system's design allows context sensing and recognition to be performed on the device. Raw sensor data do not need to be sent to the server for recognition as the process is performed on the device. This reduces the number of handshakes between the CAMR client and the CAMR server, thereby reducing response time and improving scalability. This also serves to reinforce the second requirement above, by keeping potentially sensitive user information on his device. For example, recognition of user's location, activities, etc. is realized on the device and this information is sent to the server only when needed, and after the user's permission must have been obtained. Additionally, the number of candidate items can be pre-filtered offline so that only a few number of items is used finally in the context-based processes [18]. Other alternative considered as future work in this thesis is to combine matrix factorization optimization such as singular value decomposition (SVD) with the nearest neighbor approach used in this thesis [63].

4. *Minimization of required user's interaction*: Another non-functional requirement is to minimize the number and frequency of interactions between users and their devices. In fact, any system that constantly requires user's

intervention to realize its objectives would be a burden to the user rather than being of help. In this regard, a context-aware recommendation system should not rely on the users to supply basic preference and contextual information. Provision should be included to infer user's preferences in contextual situations.

5.6.2 CAMR basic services

In this section, primary services provided and consumed by CAMR platform to satisfy the above requirements are described. First, a summary of these basic services as well as their brief descriptions, including their relationships with the use cases presented in the last section, is presented as illustrated in Table 5.17. The full descriptions of these services including other services they rely upon to fulfill these requirements are given in section 5.7.

Table 5.17 - List of CAMR services and related use cases

CAMR Service	Description	Related use cases
Context management service	This is the service responsible for monitoring, acquisition, and classification of low-level context data to infer high-level contextual information	Get contextual information
Contextual user profile management service	This service is responsible for the processing of contextual user profiles, expressing user preferences in contexts	Retrieve user preference, Generate contextual user profile and predict contextual user preference use cases
Recommendation management service	This service is responsible for the recommendation processes, which include traditional and contextual recommendations.	Get contextual recommendations and Get Recommendations use cases
Media profile management service	This service is responsible for retrieving multimedia content metadata and processing it into media item profile to be used by the recommendation service during filtering process.	Get multimedia metadata profiles use case
Presentation management service	This service is responsible for displaying recommended multimedia content in format suitable to the user's device.	Present Recommendations use case
Optional recommendation adaptation service	This is an optional service, which is provided by third party application, this explains why it is not included in the use case model.	N/A
User feedback management service	Responsible for both explicit and implicit user feedback process management	Get user feedback
External service management	Responsible for managing content or service provider services	Manage content provider and Manage service provider
Communication management	This is logical service responsible for how information is exchanged between various components of CAMR subsystems	All use cases
Client service	Client service is responsible for client side recommendation processes such as managing user's basic preferences, recommendation request	Get contextual information, Infer contextual information

5.6.2.1 Context management service

As discussed in section 5.3 (Usage scenario), classified and inferred contextual information serves as a trigger for the recommendation and filtering processes to provide personalized recommendations as presented in the use case scenario.

In this scenario, when Ana's context changes, the system is detects it by collecting low-level context data from Ana's mobile phone. Its uses the data to classify the activity that Ana is currently performing and consequently representing first stage inferred context together with the acquired low-level context in a machine processable format, such as pure XML or any other XML based descriptions such as MGEG-21 UEDs. It then sends it to the context service consumers (in this case, let us call it the recommendation management service of the CAMR server subsystem, which is presented later).

Context values extracted from these XML based descriptors are instantiated into a context inference model, e.g. ontology and rule based model; high-level context is inferred, as described in chapter three and sent to other component of the server subsystem.
An example of such inference is to infer from Ana's Wi-Fi, device GPS or CellID, the knowledge that she is at home, at university, at the train station, etc.; or deducing that it is evening on a raining weekend from low-level temporal information and weather forecast services. Additionally, it captures as first level high-level context information, Ana's activities such as *"jogging", "walking", "sitting"*, etc., from low-level context data and inferring contextual information, such as *" Ana is jogging in the sport complex"* or *"Ana is seated at home".* Thus, the context management services consists of two main components. The context recognition service, which runs on the user device to identify and classify user contexts from the device-embedded sensors. The second context management service run on the CAMR service, which obtains contexts information from the CAMR client (through context recognition service) and processes this information using ontological and SWRL based model to obtain user contextual situations. In this way, context services can be deployed on heterogeneous devices, allowing heterogeneous context consumers to access them. In addition, there is a context management service, which takes resources, such as context features produced by the client service, and processes it to provide higher-level contextual information or situations. The process responsible for obtaining this kind of information has been described in chapter three. However, in this chapter, we will analyze software system implementing this context recognition service. The contextual features such as "weekend", "sitting", "home", etc., are provided as resources that external sources can consume via a Web service interface. The proposed system is designed to provide context monitoring and recognition as a service that can be implemented on the user's devices, providing such service in a timely and interoperable manner.

5.6.2.2 Contextual user profile management service

A user profile summarizes the preferences of a user, normally based on the history of the user's action. As discussed in chapter 4, in our framework, it summarizes user's consumptions into a limited set of categories. Categories are characterized by a set of genres; a number of properties in turn characterizes these genres. One or more genres could be associated with each category. Several properties may be associated with one genre. Additionally, it incorporates the contextual dimension, associating one or more inferred context with each *category-genre-property*. Accordingly, the contextual user profiling service generates a contextualized user profile based on the inferred context, thus obtaining the preferences of users in each specific context. This service is also responsible for learning user preferences, using the user's contextual situations.

In the case of Ana in the scenario, her preference for pop or jazz music could be learned by knowing contexts, such as *"jogging"* or *"walking"* in which she has consumed that kind of content in the past. The contextual user profiling service provides each user profile as a resource, accessible via a Web service interface. To ensure that the system keeps up with the user's preference changes, the user profile service updates user profile in two forms: explicitly and implicitly. The former grants users the ability to indicate, using numerical values the level of satisfaction they derive from the recommendations provided for them. The implicit approach involves preference learning through implicit feedback. The profile is updated implicitly whenever a user consumes or decides not to consume any kind of media item provided as recommendation. A context agent of the mobile phone platform collects relevant contextual data from available sensors (accelerometer, GPS, rotation and orientation sensors, etc.) as well as from the network connectivity. The system analyses these values to infer that the user is in a specific context, such as "sitting at home".

Once the user's context is inferred, and the consumed content is characterized, the user profile update process is performed. The system compares the media metadata with the list of properties it uses. Every match will trigger a search into the current user profile (if he or she is an existing user). If the user is new, a complete new user profile is created based on his current contextual situation. The system uses this information with data about recommended media items to learn the consumption preferences of the users by measuring the "intensity" of his consumption of a particular content in specific contexts. For explicit profile update, a provision should be made to allow users to assess the recommended content directly. On the other hand, provision should be made for an agent to monitor users

interaction with the system, which reports user's consumption choices, and uses it to update the user profile according to what has been consumed or not in a specific context. Details of the relevance feedback and the user profile update process are provided in chapter four.

5.6.2.3 Media profile and recommendation management

The context-aware recommendation service explores three modes of operations for contextual media recommendations. The content base (CBF), the collaborative-based (CF) and hybrid based operational modes. In the context-aware content based mode, after obtaining Ana's high-level contextual situation in the scenario above, the system searches via the contextual user profile service, to find if Ana has been in the same or a similar contextual situation in the past. If Ana has never been in a similar contextual situation or if Ana happens to be a first time user, in either case, there is no adequate context history or preference information available to the recommendation service to utilize. Thus, the system can switch to context-aware collaborative mode to address this kind of situation by, for example, using the contextual history of other existing users who have been in contexts similar to Ana's current context. In the context-aware hybrid mode, the *new user* problem is also handled by a combination of context-aware collaborative filtering and content recommendation approaches. For an existing user, the system could use any of the three recommendation services depending on the inferred contextual preference. From the inferred preference, the category of content preferred is determined. If the category of content that would interest the user in the present context were Movie, for example, the contextual collaborative process would be selected. If it suggests "news", contextual content-based process could be selected as the ideal recommendation service. Context and activity information are explored to find a match between currently inferred high-level context/activity and those previously stored in the user profile. Thus, filtering process is achieved in two alternative phases as follows:

(1) A match is found

If the system finds a match (i.e. If Ana had already consumed something in that context or similar context), it goes ahead and executes processes to predict Ana's contextual preference in her current context and selects appropriate recommendation process accordingly.

(2) *A match is not found*

If the system does not find a match (i.e., if Ana is a new user and has never consumed anything in that particular context), it searches the entire user profile repository looking for users who have been in a similar context as Ana's present context. It uses information about the content consumed by those users to recommend content to Ana. If no user has ever consumed any content in such context, then it ignores the contextual information and builds a completely new user profile for Ana.

The media profile management service obtains the multimedia content metadata from external sources and processes this information for each media item into what we call media item profile used in the recommendation process to compare the similarity between users and multimedia items.

5.6.2.4. Recommendation presentation service

This service is responsible for the presentation of recommended items to users. Because the proposed recommendation service framework is designed to provide diverse categories of content in different formats, such as in audiovisual and textual forms, this service ensures that the content being recommended is presented to the user in the best possible format in the present contextual situation, by providing alternative presentation formats. For example, in the scenario, the system presents Ana with a filtered and personalized list of recommended media items, from which she selects one or more specific item to display. Ana has the options of viewing the recommended movie items in audiovisual or in textual form, depending on Ana's current context or activity. This service is designed to work with the adaptation service, which is responsible for automatically deciding the best possible format of recommendation presentation, according to contextual constraints presented by the device characteristics and network conditions.

5.6.2.5. Recommendation adaptation service

The indication of the selected item is (optionally) forwarded to an adaptation decision engine to check if its format satisfies the constraints imposed by the current usage context, especially the device's characteristics and network conditions. If it does not, the item is subjected to adaptation. In the scenario, an important observation is that the system should be able to use contextual information to adapt the presentation of the recommended content. For example, the system should possess the ability to present synopsis of recommended movie items when it observes that "Ana is walking home from school", but "when she is at home", the device, then provides her with the movie trailers, and with URL links to watch the full movie if she chooses. On the other hand, if the system discovered low battery problem, it could possibly perform a summarization of the video clip. However, details of the design of such low-level adaptations are beyond the scope of this thesis, thus it assumes the availability of an adaptation service such as MULTICAO to execute this content adaptation [127].

5.7 CAMR detailed functional architecture

5.7.1 CAMR layered architecture

The scenario, requirements and services as presented in the previous sections provide the impetus for the development of CAMR functional architecture. The functional architecture of a system generally describes a collection of components that make up the system and how those components interact in order to realize the system's functionality, especially to fulfill its basic functions. In this regard, we have extended the conceptual architecture presented in Figure 4.2 of chapter 4 as depicted in Figure 5.5, showing more details and depicting its constituent components in a layered form.

The architecture roughly comprises three important layers, which have been designed to realize the functional and non-functional requirements of the proposed system as presented in the last section. The first layer constitutes applications that are developed to profit from the services provided by CAMR software system. These include third party applications of service providers or content providers who use CAMR services to personalize their offered services to their consumers. We regard this layer as CAMR client subsystem layer. It comprises those components such as the context information related components, user preference setting, and client applications, such as personalized recommendation client applications, that run on the user devices. The direct beneficiaries of these services are service consumers, i.e. the mobile users who use diverse kinds of mobile devices to consume multimedia content. The second layer constitutes the middle layer. This layer contains the core components of CAMR, which run on the CAMR server subsystem.

These components are responsible for user profile management, contextual recommendation management, media profiling, etc. This layer also comprises domain models and their components with facades for realizing other core functional requirements of the system as specified in section 5.1.
The third layer is the persistence and knowledgebase layer having components that abstract the logics required to manipulate data such as user profile information, contextual history, and other data contained in the knowledge base. These components can utilize the existing data access framework such as Object/Relational Mappings, or persistence APIs for easier CRUD (Create, Retrieve, Update and Delete) operations of the various components of the system.

5.7.2 Detailed descriptions of CAMR services

The proposed system is designed to interact, via Internet, with various online-based media item providers, users, and other service providers as illustrated in Figure 5.6. The architecture shows in detail various functional parts that realize the values delivered by CAMR system. It also shows, in addition to the actors, the movement of message from one functional component to another. The orange colored functional component, **Media Adaptation Management Service** is not directly implemented by the CAMR framework. It is a service that can be consumed from an existing system such as MULTICAO [153]. Therefore, for example, the architecture assumes that media items are provided with open standard descriptions, based on MPEG-21 UEDs [6] and MPEG-7 MDS [48] or Web service via public

APIs. The standardized descriptive metadata provides detailed information on the media items, such as their characteristics, e.g. genre, media types, and other low-level characteristics such as frame rate, bit rate, etc.

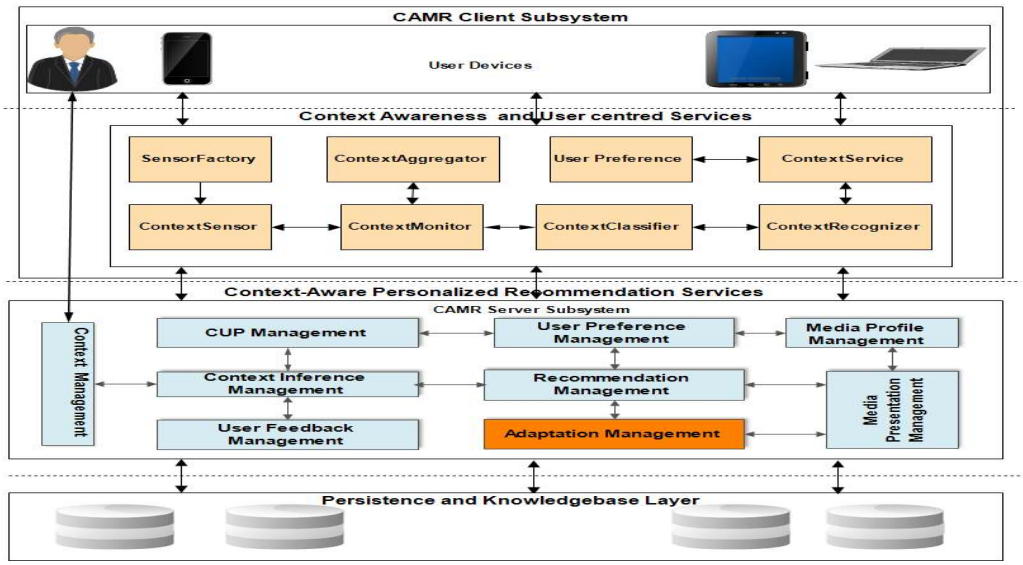


Figure 5.5- CAMR layered architecture.

For this reason, the framework was designed to allow interoperable interactions between its components and other external entities. Thus, the framework's design relies on object-oriented design principles such that objects with similar functions or with supporting functions are grouped into components and modules. The functions are then exposed as web services. The services can be realized based on REST architectural style [183] or SOAP [184]. Nevertheless, the current implementation relies on RESTful web services. Additionally, it also needs to provide contextual information such as device characteristics, contextual user preferences, user activities, usage environment conditions, network characteristics. This information is provided and is communicated by different heterogeneous components of the framework. The communications between these components are stateless; thereby every request from a client component to the server component contains all necessary information needed to understand the request. Therefore, sessions are kept entirely on the client component.

This improves scalability and reliability because the server component (service provider) does not have to manage resource usage across requests, thus quickly freeing up resources and easing recovery from partial failure [183]. In this section, an overview of different aspects of the proposed system as depicted in Figure 5.6 is provided. As can be seen in this figure, all the modules interact via Web service interface. Each subsystem of the architecture comprises components, which realize one or more of the functionality or cases elaborated in the last sections. These functionalities are provided as services, thus, the architecture could be regarded as a service-based architecture or Service Oriented Architecture (SOA) on one hand and Resource Oriented Architecture (ROA) on the other hand. The architecture is resource oriented because services provided namely context awareness, recommendations, user profiling, etc. are modelled as resources that can be consumed by other external services. The architecture integrates the system's components to support each of its functionality by organizing them into specific areas of concerns. REST is a representational state transfer style, which abstracts architectural elements within a distributed system. Exchange of messages between each component and external entities relies on REST web services. REST, rather than focusing on component implementation details, focuses on the functions provided by the components, the interactions between them, and external entities as well as the interpretations of their data elements.

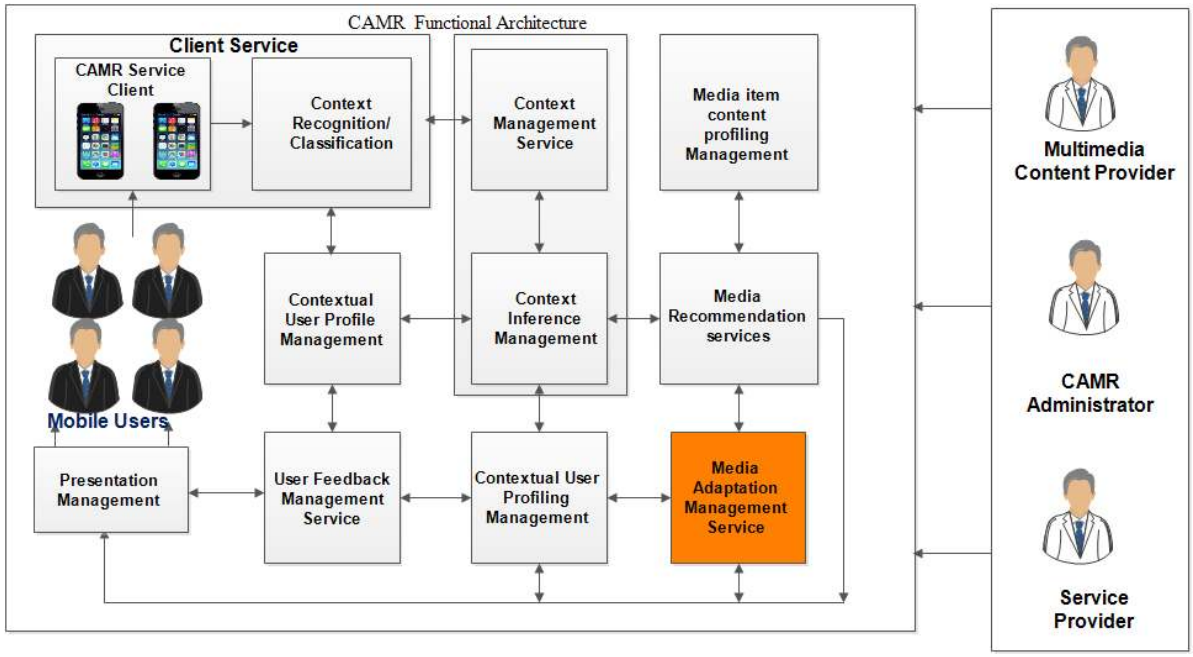


Figure 5.6- CAMR functional architecture.

Thus, the proposed architecture is designed to offer support for dynamic and implicit context-aware personalized media recommendations in mobile environments, in a flexible and interoperable way. At a global level, as earlier stated, the architecture conceptually consists of three-layered subsystems, providing and consuming services. For example, the client subsystem layer provides context information as services and can be deployed on any device of any hardware or operating system platforms, such as laptop, smartphones (iOS, Android, Windows 8, etc.) and even desktops. The server subsystem layer comprises the main services responsible for the main functionality of the systems. These include Recommendation Management(RM), Media Profile Management (MPM), Adaptation Management(AM), Contextual User Profile Management (CUPM), Context Management (CM), Context Inference Management (CIM), User Feedback Management (UFM), Communication Management, and Presentation Management(PM).

The other layer consists of the data and persistent component utilities for accessing, manipulating, and storing usage history, knowledge bases, user contextual preference persistence, and contextual information storage. Among these services, the system ensures that non-functional requirements are met. For example, it ensures that power consumption by the user device is optimized, especially during context monitoring, capturing, classification, and recognition. In addition, it is meant to ensure scalability and privacy/trustworthiness of the system are realized. In the next sections, these services are discussed, presenting their designs, following software engineering principles. In terms of implementation, the concrete implementation of the architecture can be realized using typical client-server architecture.

5.7.2.1 Recommendation management

The recommendation management is responsible for two important functions of the architecture. First, it implements filtering processes, which use the contextual user profiles and the media item profiles generated by both contextual user profile and media profile management services respectively described in the next subsections. In the recommendation phase, the recommendation management service takes as inputs the contextual user profiles and the

media item profiles and produces as output a list of media items as recommendations. The final list of recommendations is sent to the presentation management, which determines the best form in which each of the recommended items should be displayed to the user. The recommendation manager's second function is the coordination of the entire recommendation process, which in this platform can be executed by one of three filtering services. First, context-aware content-based recommendation service filters and provides recommendations based on the user's consumption history considering the contextual situations in which such consumptions were made. Second, the context-aware collaborative recommendation service filters and provides recommendations based on the consumption of users who have consumed items in contextual situations similar to the current contextual situation of the candidate user. Because of the limitation of these two filtering services, a third service, the context-aware hybrid recommendation service is provided. The algorithms employed by these filtering services have been presented in chapter four. The recommendation management also coordinates other services provided by the system, such as contextual user profile management service, context and inference management services, adaptation management service, user feedback management service, and presentation management service. Accordingly, this component of the system can be seen as the overall system orchestrator.

5.7.2.2 Media profile management

The media profile management is responsible for implementing multiple functions. First, it is responsible for crawling metadata of candidate and consumed media items from external sources such as online-based content providers. For example, in the case of movie and music items, metadata of candidate movies and music items is obtained from databases with public APIs such as IMDB, TMDB and Last.FM. The media profile management also parses the candidate media items metadata represented in standardized formats, namely MPEG-7 MDs, etc. and stores them in the framework data stores. The metadata representation is structured in three classes. First, the description of the candidate media item, which includes its physical properties such as location URI and media format. The second class encompasses the content description, such as the title of the media item, its genre, its category, etc. The third class includes the contextual description of the media item such as its creation date, its play duration, language, scope, etc. This component also updates the media metadata storage with the latest descriptions of candidate media items. This service, in conjunction with the contextual user profile management, filters candidate media item metadata based on the contextual user preferences and produces a representative profile of each candidate media item in a form that can be utilized by the recommendation services to provide a set of recommended media items. The media profile management also assists the user feedback management and contextual user profile management during the user profile update process. The metadata of items consumed by the users in specific contexts are provided during relevance feedback and the user profile updated accordingly.

5.7.2.3 Adaptation management

The adaptation management is an external service responsible for the recommended media item presentation adaptation. The service is functionally responsible for adapting media item contents based on summarization and transcoding techniques [153]. The adaptation management could adapt the content such as audio or video according to the battery power remaining, or due to network bandwidth constraints. Transcoding would transform the contents from one media type to another based on the network conditions or even according to the device capabilities, such as screen size or codecs installed. Because this service is an external service, its design and implementation are beyond the scope of this thesis. Andrade et al. [127], [153] present a very good example of the adaptation service that can be incorporated into the CAMR framework

5.7.2.4 Contextual user profile management

The user profile describes preferences, normally based on the history of the user's actions [6]. CAMR's contextual user profile summarizes the user's content consumptions into a limited set of categories. Categories are characterized by one or more genre, and a number of properties characterize the genre. Several genres can be associated to one category. Several properties can be associated with one genre. Additionally, it incorporates the contextual dimension,

associating one or more inferred context to each *category-genre-property* concept. The contextual user profile management component implements all processes related to the user profile, notably by implementing user profiling algorithms described in chapter 4. It is thus responsible for creating and updating the user profile and for filtering it according to contextual conditions thus generating the contextualized user profile that is used for the recommendation processes. Additionally, in order to improve the quality of content being recommended to users, it works with the user feedback management to update user's profile information via relevance feedback obtained whenever users interact with the system and consume the recommended content.

5.7.2.5 User feedback management

In the architecture, it is important to keep track of user's consumption behavior to improve the system's recommendation quality and accuracy. The user profile can be updated based on two well-documented approaches, explicit and implicit methods, which are integrated in the User Feedback Management processes. The former grants the users the ability to modify the values assigned to their preferences by the system. The implicit approach involves preference learning without direct user intervention, such as updating the profile when the user has spent a certain amount of time on a given item. The CAMR's user feedback management (UFM) service updates the user profile whenever the user interacts with the system; it tracks both consumption and non-consumption of content by the user to learn the contextual preferences of the user for any kind of media item. To learn the user preferences implicitly, UFM runs an intermittent background service monitoring the interactions of the users. Additionally, it explicitly updates the profile whenever a user consumes any content item by obtaining a feedback from such user. Details of relevance feedback based user profile update process has been presented in chapter 4.

5.7.2.6 Presentation management

The presentation management service is responsible for the display of recommended items to users. It works with the recommendation management to determine the best suitable format to display the recommended media items. Additionally, this service works with the feedback management service, by providing an interface where users can provide explicit feedback on items they consume. For example, when recommending movies to users, based on the current contextual situation of the user such as when she is at home on Friday morning, the presentation management may decide to display only the synopsis of the recommended movies with the link to play the trailer of such movies. It may also decide to play the full movies via online repository to watch the movies (provided the user has subscription to such services) if for example the user is at home on Friday or Saturday night.

5.7.2.7 Context management

The context management implements the server side functionality conceptualized and described in chapter 3 as part of the CAMR. It interoperates with the context-related modules at the client side instantiating the contextual data, gathered and partly processed and classified by the client into the knowledge-based mechanisms to infer additional high-level knowledge. The context management receives and parses the contextual features from the client based context recognition module and instantiates them in context inference management module. The context management module implements the mapping of context features to the concepts and objects of the ontology model. It also implements rule based mechanisms for inferring semantic meanings from context information by establishing relationships among contextual concepts, leading to higher contextual information. The higher-level contextual information obtained from context management modules can be used for two purposes. First, it can be utilized directly in the recommendation of media items that match user's contextual needs. Second, it can used in the user profiling process, especially, for predicting user's contextual preferences, which are consequently used to provide contextual recommendations to users. For example, if a user has preferred in the past to watch romantic movies in rainy weekends, this information could be used to infer his preference in a dry weekend, i.e. non-raining weekend. Additionally, this module represents the classified context features in a standardized description suitable for interoperable transmission to other components of the architecture, especially the main context management module on the server subsystem.

5.7.2.8 Communication management

The communication management (COMM)) is responsible for managing the communications between various services of the architecture. The COMM is also responsible for the transmission of resources across network interfaces. Since the architecture is distributed, various services can be in different locations (even geographically) and the services can interact regardless of their physical distance. For example, the communication of contextual information from the context recognition management on the client subsystem and the recommendation management service on the server subsystem. There are two variants of the service. The first is the communication service on the client subsystem, which is responsible for communicating contextual information and user feedback information from the client subsystem to the server subsystem. The second service is the request dispatcher service, which runs on the server. It is responsible for handling communication between local services as well as handling the external communication of the server: with the client subsystem or with other external sources such as online-based service and content providers. These two communication services were implemented as REST services on top of HTTP in two modes. First mode is the pull mode, which allows services that need information to query the service providers and wait to get the response (request/response communication model). The other communication mode is the push mode, in which service consumers subscribe for a piece of information and get notified whenever such information is available (subscribe/notify mode). In addition, this design can profit from existing application programming interfaces, such as Java Messaging Services (JMS) or SOAP with Attachments API for Java (SAAJ) [185]. Nevertheless, there are other implementation options, depending on the implementation platform.

5.7.2.9 External service management

The external service management implements and maintains the external APIs for accessing external services, such as crawling and parsing multimedia metadata from online sources. Essentially, this service works with the Media profile management service to provide metadata of candidate media items, which is crawled from various online-based databases.

5.7.2.10 Client service

The client service makes request on behalf of the mobile users whenever the context recognition service observes significant changes in context information or when users explicitly request for recommendations.

Additionally, it can be used to make recommendation request based on intervals of time, i.e., periodically making recommendation requests. Another function of this service is based on the client subsystem is the user information and preference service. This service manages sensitive user information and preference on the user's devices. Rather than keeping such trusted information on the server subsystem, this information is optionally provided by the user and stored on their devices and will be communicated to the server subsystem via the communication management service only on user's approval. Finally, the architecture also includes those services that are responsible for ensuring certain crosscutting concerns such as trust, security, and trustworthiness of users on the entire systems. It is also responsible for certain non-functional requirements such as scalability, and timely provisioning of recommendation with minimal user direst interventions.

5.7.3 CAMR sequence model

In this section, we present and discuss the operations and interactions of the components of the system's functional architecture. We have chosen sequence diagrams to present these interactions and activity models of the CAMR system. The sequence diagrams specify how the framework's components interact to accomplish its functional requirements. They show, at a higher level, the most important interactions of each component, representing core sequences of interactions between those components of the system as it operates. The sequence models capture the order, in a timely manner, of how the interactions between these components are triggered to realize the use cases or the functionality of the proposed context-aware media recommendation framework. Six key operations of the architecture are modeled using sequence diagrams, which logically abstract the functional requirements and use cases

presented in the previous sections. Table 5.18 presents the sequence models and associated use cases as well as their brief descriptions. These models represent the interactions as presented in the following section. The interactions are established between software packages constituted by a number of classes. Those classes and packages are described in sections 5.7.4 and 5.7.5.

Table 5.18 Sequence models and associated use cases

Sequence Model	Description	Associated use cases
Context recognition and classification sequences	It describes the interactions when system obtains raw sensor data, aggregates, filters, extracts features, and applies a classification model to obtain higher-level contextual features in the client subsystem, and the semantic context inference at the CAMR server subsystem	Monitor Context Information , Get Contextual Information, Infer high-level contextual information
Determination of contextual preference sequences	This sequence involves using user's current contextual information, which is obtained from the context recognition process to determine automatically the user's preference in that context.	Retrieve Preference, Monitor Context Information , Get Contextual Information, Infer high-level contextual information
Relevance feedback and contextual user profile update sequences	Describes the operational sequences of the user feedback and user profile update process. The interactions in this model capture what happens after users have been presented with contextual recommendations, especially, how it uses the knowledge of user's reactions to the provided recommendations to learn about user's taste in specific contexts in order to improve future recommendations.	Retrieve Preference, Monitor Context Information , Get Contextual Information, Infer high-level contextual information
Context-aware recommendations sequences	This sequence describes interactions that occur either from when explicit or implicit recommendation process is initiated to the time the user obtains a list of recommendations.	Retrieve Preference, Monitor Context Information , Get Contextual Information, Infer high-level contextual information
Contextual user profiling sequences	It describes the sequences of interactions between the components of the system when building the contextual user profile. These sequences capture how contextual information is used to build the contextual user profile, which is used to compute the contextual user profile vector, the representation of the user's dynamic preference, based either on past contextual preferences or on the contextual preferences of like-minded users.	Obtain Explicit User Feedback, Obtain implicit User Feedback
recommendation adaptation sequence	Describes interaction of the component of the system during recommendation process	Get contextual recommendations and Get Recommendations use cases

5.7.3.1. Context recognition and classification

The basic sequence of interactions that occur during context recognition and classification process is illustrated in Figure 5.7, showing how the system obtains raw sensor data, aggregates, filters, extracts features, and applies a classification model to obtain higher-level contextual features in the client subsystem. In Table 5.19, we present brief descriptions of some important messages exchanged during this interaction sequence. The inferred contextual feature is delivered via the *CommnicationManager* (implementation package of communication management) to the server subsystem. The server subsystem then uses the received context information to trigger contextual recommendation processes. The classified context information is represented and delivered in XML based standard formats such as MPEG-21 UED for interoperability support between the system's modules as earlier stated. First, *ContextMonitors* at predetermined intervals start the *ContextSensors* by sending a *startSensor()* message to the *ContextSensors.* Thus, the main function of the *ContextSensors* in this interaction is to retrieve events being generated by each hardware or software sensor via their respective APIs.

Thus, the *ContextMonitors* collect sensor events and send them to the *ContextAggregator*. The *ContextAggregator* collects the raw contextual events/data from the context monitors, and aggregating the events from one or more sensors. It then performs a number of operations, such as preprocessing of the sensors' raw data, using the *preProcessRawData()* method and filtering using the *filterRawSensorEvents(),* by taking the events as inputs, and by removing potentially erroneous events from the streams. It then calls the *extractFeatures()* method for feature extraction with the filtered events as inputs, and returns important features capable of discriminating between high-level context. It then prepares the event streams as inputs to a classification model implemented by the *ContextClassifier* class for the recognition of the labeled high-level context information. The high-level contextual information that corresponds to the low-level sensor data via its *classify()* and *executeClassification()* methods. The *ContextClassifier* class produces higher-level contextual information such as user activities and other contexts from the raw context features is returned.

Table 5.19 - Selected functions of the context recognition and classification sequence

Method	Function
startSensors	This method activates the sensors in order to begin data collection process, it has a corresponding stop version that stops the sensor anytime it is desired either by the user or at intervals set by such user.
sendContextEvents	This method collect event from sensors and prepare the events for preprocessing
filterRawSensorEvent/preProcessRawData	This method implements algorithm for filtering sensor data
extractFeatures	Implements feature extraction process
classify	This is the method that implements context classification algorithm(s)
executeClassification	It executes specific classification algorithm
recogninizeContext	This method produces the high –level labeled context information after the execution of classification algorithm

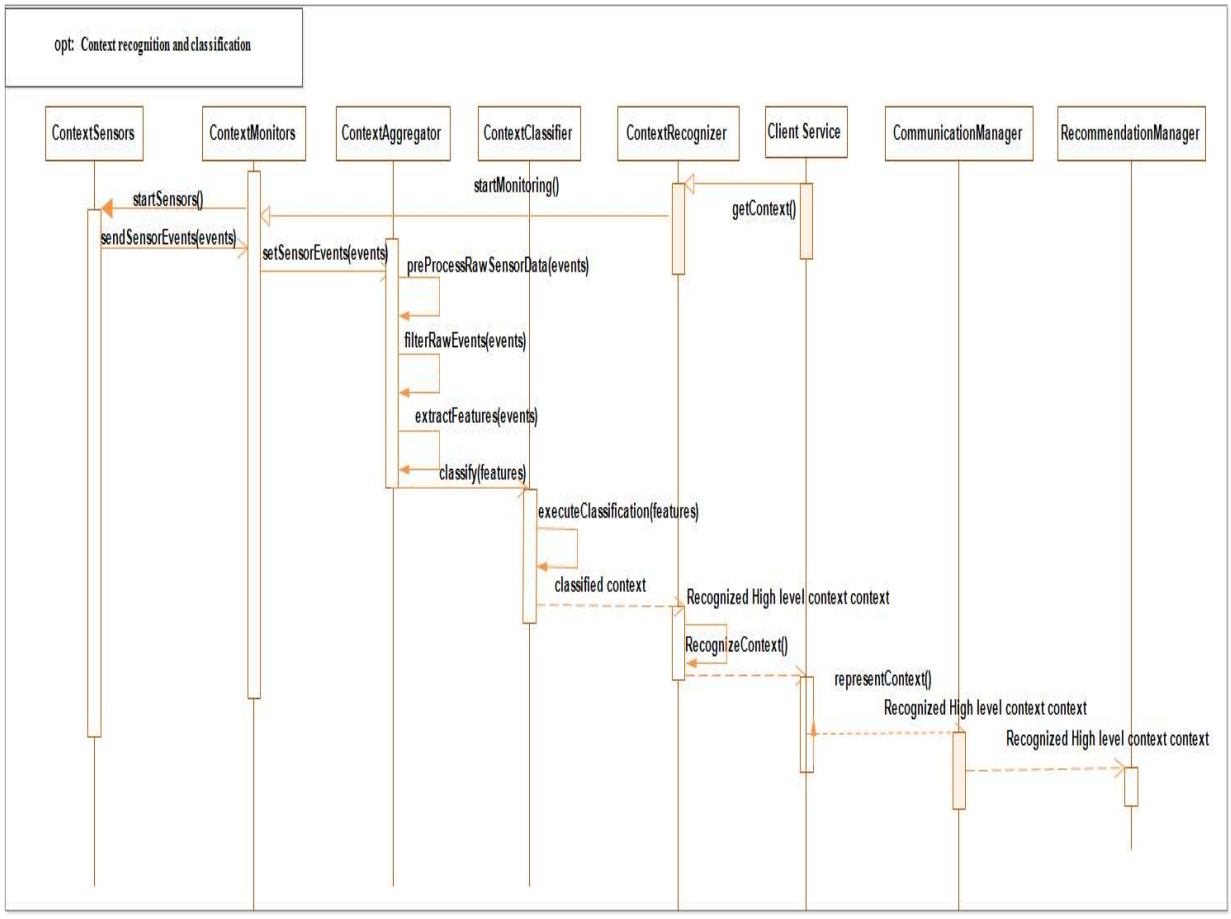


Figure 5.7- Simplified sequence of interactions in context recognition

5.7.3.2. User profiling and preference determination sequence

The contextual user profiling and preference sequence, illustrated in Figures 5.8 and 5.9, involves using user's current contextual information, obtained from the context recognition process to determine automatically the user's preference in that context. Table 5.20 presents brief descriptions of messages exchanged during this interaction. Having obtained the contextual information as well as user basic setting on the device via the *CommmunicationManager* from the client subsystem, the *RecommendationManager* (implementation package of recommendation management) initiates the preference determination process by sending *inferUserPreference()* message, including the user context information as well as user preference as its input parameters, to the *UserPreferenceManager* via the *UserProfileManager* (implementation of the contextual user profile management) module. The *UserPreferenceManager* then creates a *reasoner,* which determines the corresponding user preference. The reason behind using a *ReasonerFactory* is to allow flexibility allowing implementing other reasoning algorithms and exposing them through common interface. For instance, nearest neighbor algorithms, Bayesian network, etc. can all be implemented as *reasoners* to infer the user's contextual preference. The *ContextManager* instantiates the context information into the inference model via *ContextInferenceManager* and processes the context information to infer user's contextual situation. The *UserPreferenceManager* then processes the request, using the user's high-level contextual information and user profile information as input parameters and returns the preference to the *RecommendationManager*. Accordingly, the model in Figure 5.9 illustrates sequences of interactions between the modules of the system responsible for contextual user profiling.

These sequences capture how contextual information is used to build the contextual user profile, allowing computing the contextual user profile vector, the representation of the user's dynamic preference, based either on past contextual preferences or on the contextual preferences of like-minded users. This interaction is initiated by the *RecommendationManager*, which receives user contextual information from the client subsystem via a *requestRecommendtionProcess()* with user's contextual information and preference setting as input parameters. The *RecommendationManager* then starts context user profiling process by sending an *inferUserPreference()* message to the *UserPreferenceManager* via the *UserProfileManager, a*s explained in the contextual user preference sequence model section.

In this process, the *RecommendationManager* must determine if the user is a new or a returning user. A new user is one who has not used the system before and thus does not have adequate profile information to assist *RecommendationManager* in the recommendation process. On the other hand, a returning user is one that has used the system before and has past preferences and contextual information that could be used to predict his preference in the present contextual situation. If the user is a returning user, the *RecommendationManager* sends a message to retrieve the user's past contextual preferences and then initiates the *retrieveUserPastContextualPreferences()* on the contextual profile class to obtain the user's past contextual preferences.

After obtaining the past user preferences, the new preference is then used to build to filter the old preferences via *filterContextualPreferences()* message to the *ContextualUserProfile,* which then builds a new contextual user profile using the *buildContextualUserProfile()* process, and the new contextual user profile is returned to the *RecommendationManager.* If the user is a new user, a slightly different process is initiated. Rather than using the user's past preferences, first, users who have been in similar context as the user's current context are determined These are the so-called neighbors of the user. This process is realized via the *getNeighborsPastPreference()* and their preferences in those similar contexts. These preferences are used to build a contextual user profile for the new user.

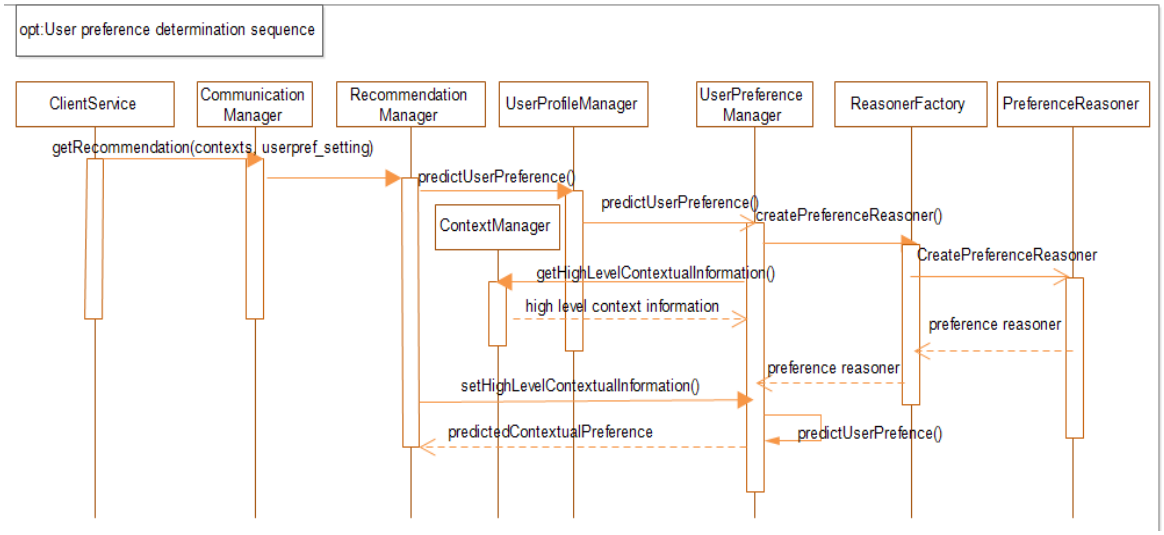


Figure 5.8- Simplified sequence of interactions for user preference inference

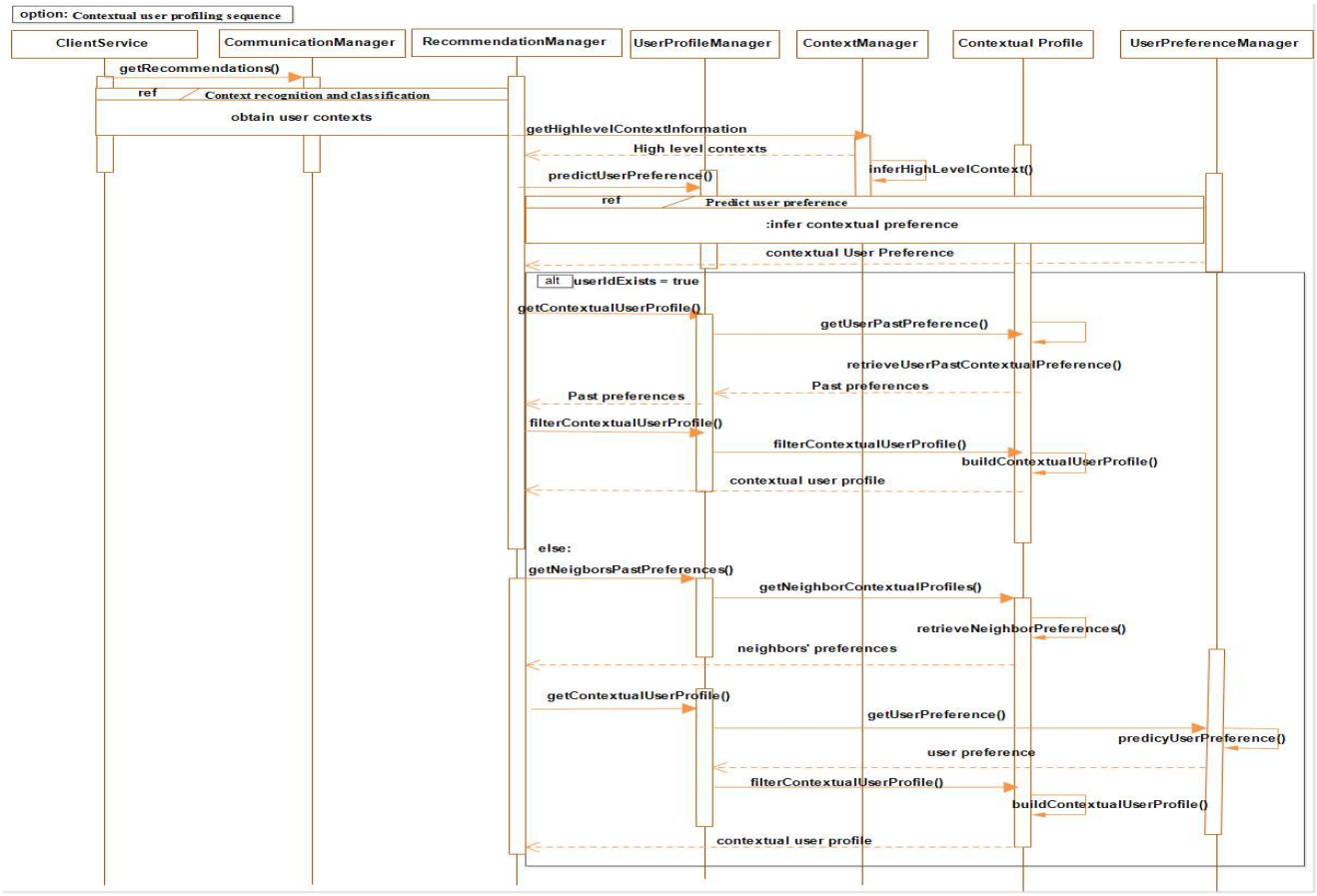


Figure 5.9- Simplified sequence of interactions during contextual user profiling.

Table 5.20 - Selected functions of the user profiling sequence

Method	Function
getRecommendations	This function is initiated when there is context change or when user explicitly makes a request for recommendations
inferHighlevelContext	Obtains user's high-level contexts
predictUserPreference	This method executes the algorithm for determining user preference in the present context
getContextualUserProfile	This method delivers user's profile containing the user preferences in relation to the contextual information
filterContextualUserProfile	This method implements the algorithm to determine user's preference in specific context.
buildContextualUserProfile	After determining user's contextual preference, a new contextual user profile is processed to reflect the newly predicted preference.
getNeigborsPastPreferences	In contextual collaborative recommendations, this method assists to obtain the preferences of friends of the target user
getNeigborsContextualProfiles	Assists to retrieve the profiles of the friends of the target user
retrieveNeigborPreferences	Assists to obtain past preferences of the friends in specific contexts
getRecommendations	This function is initiated when there is a context change or when user explicitly makes a request for recommendations
predictUserPreference	This method executes the algorithm for determining user preference in the present context
createPreferenceReasoner	Because it is possible to have more than one algorithm to determine user's contextual preference, this method creates specific reasoned for this purpose.

5.7.3.3. Recommendation sequence model

Having received the contextual user profile, either for the returning or new user, the *RecommendationManager* then continues with the recommendation process, as depicted in Figure 5.10 with Table 5.21 describing briefly the messages exchanged during this sequence of interactions. First, the *RecommendationManager* determines the category of content that candidate media items to request from the external sources belong based on the contextual preference of the user. Then, it sends a *getCandidateMediaProfiles()* message to the *MediaProfileManager* (implementation of package of media profile management), which then obtains the candidate media item metadata in standardized representations, such as XML or Json format from external sources, especially from the Internet-based media repositories. The format could also be in MPEG-7 MDs. The assembled metadata then undergoes a filtering process via *filterCandidateMediaProfile()* method of the *MediaProfileManager* module, to produce the media item profiles. Note that the *RecommendationManager* decides on the appropriate recommendation service to invoke after obtaining user's preference before executing recommendation process via the *RecommendationEngine*. For example, depending on the category to which the candidate media items being processed belongs, it would determine *via createRecommendationProcess()* method, to trigger one of context-aware recommendation processes.

If the content category were movie, for instance, context-aware collaborative filtering or a context-aware hybrid service would be invoked. After candidate media profiles have been filtered and returned to the *RecommendationManager* then *computeContexualUserProfileVector()* process is invoked to generate the user profile vector, which is then used in the *computeMediaProfileVectors()* process.
The computed vectors serve as inputs to the *SimilarityMatrix* class to compute the similarities between the user profile and each candidate media item vector. After these similarities have been returned to the *RecommendationEngine*, the predicted preference (a real value) for each candidate items is then generated using *computeRelevance()* method. The *RecommendationManager* then ranks the computed values in descending order of magnitude in the *generateRecommendations()* to select the first *n* items with the highest values. This first *n* items, as the recommendation set is then sent to the *PresentationManager*, which prepares the recommended media items, with the appropriate viewing formats for display to the user.

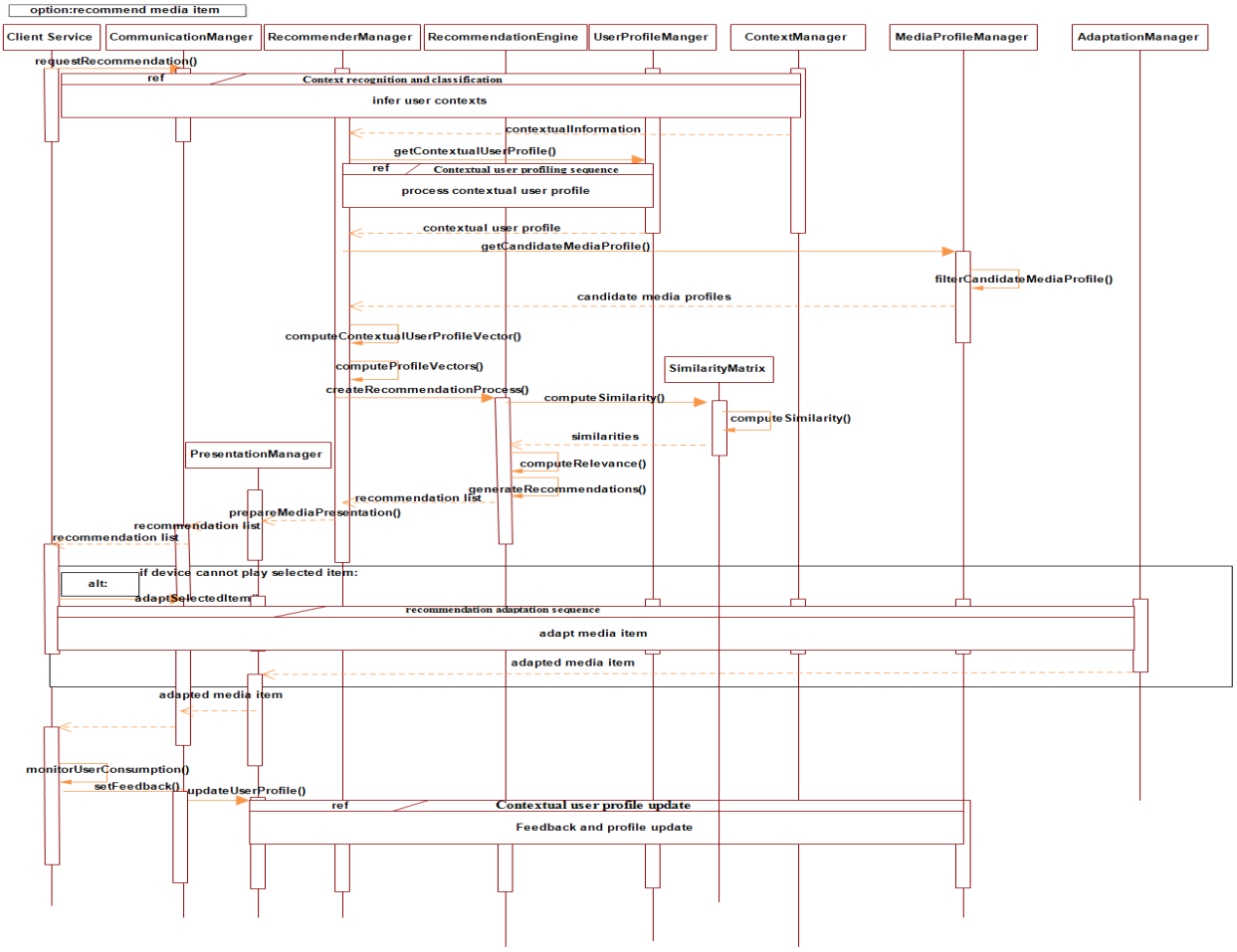


Figure 5.10- Simplified sequence of interactions for an adaptive context-aware recommendation process.

Table 5.21 - Selected functions in of the recommendation sequence

Method	Function
Please note that all previous methods in other sequences are part of the recommendation processes. We only describe methods that are unique to the recommendation process to avoid repetition.	
createRecommendationProcess	This is a factory method that creates relevant recommendation process depending on the contextual situation of the user
determineRecommendationProcess	This method determines the appropriate recommendation process based on the user contextual situation initiate the recommendation creation process
generateRecommendations	It delivers user's profile containing the user preferences in relation to the contextual information
executeContentFiltering	This method implements the algorithm to determine user's preference in specific context.
processContextualUserProfile	Obtains the user profiles for user preference determination via the user profile manager
updateUserProfile	Assists to update user profile
presentRecommendation	Assists to obtain past preferences of the friends in specific contexts
prepareMediaPresentation	This method prepares the recommended item for presentation to the user
computeMediaProfileVectors	This method is called via media profile manager to obtain each candidate media item vector
computeUserProfileVectors	This method computes for each user, contextual user profile vector via user profile manager
computeRelevance	This method computes the relevance between target user and candidate media item using nearest neighbor algorithm

5.7.3.4. User feedback and update sequences.

Figure 5.11 illustrates the simplified operational sequences of the user feedback and user profile update process while Table 5.22 presents the description of some important messages during this interaction. The interactions in this model capture what happens after users have been presented with contextual recommendations, especially, how it uses the knowledge of user's reactions to the provided recommendations to learn about user's taste in specific contexts in order to improve future recommendations. This process is executed either explicitly or implicitly. In the explicit mode, provision is made in the presentation interface for user to rate item they consume, that is items considered as relevant. Nevertheless, this process is optional, users could choose not to rate if they so wish. Thus, the second mode is the implicit mode where an agent monitors the consumption of the users and retrieves important information about the content consumed and contexts in which such content is consumed.

Table 5.22 - Selected functions in of the user profile update sequence

Method	Function
Please note that all previous methods in other sequences are part of the recommendation processes. We only describe methods that are unique to the recommendation process to avoid repetition.	
monitorUserConsumption	This is a factory method that create relevant recommendation process depending on the contextual situation of the user
getUserFeedback	This method determines the appropriate recommendation process based on the user contextual situation initiate the recommendation creation process
updateUserProfile	This method delivers user's profile containing the user preferences in relation to the contextual information
createUserFeedback	This method implements the algorithm to determine user's preference in specific context.
computeRelevanceFeedback	Obtains the user profiles for preference determination via the user profile manager
executeUserProfileUpdate	Implements the update user profile process
startUserFeedbackProcess	Assists to obtain past preferences of the user in specific contexts

It then sends this information via the *CommunicationManager* to initiate the user feedback and update process. By default, the client services continuously monitors user's consumption via *its monitorUserConsumption()* process.
Anytime there is a consumption, a message containing information about such consumption is sent via the *CommunicationManager,* which prepares the feedback information via an *addUserFeedback()* method to the *RecommendationManager*. *RecommendationManager then sends an updateUserProfile() requests to the UserFeedbackManager,* which then creates a user feedback process via the *UserFeedbackFactory's createUserFeedback()* method. The *UserFeedbackManager* starts the user profile update process via *startUserFeedbackProcess()* method and then sends a *getUserProfile()* message to the *UserProfileManager* to obtain the contextual profile of the user whose profile is being updated. It then sends a *getMediaProfile()* to the *MediaProfileManager* to obtain the profile(s) of the media items consumed by the user. When the *UserFeedbackManager* receives the profile information of the items and of the user, it then executes the user feedback process via *computeUserFeedback()* process. The output of the *computeUserFeedback()* is sent in a message to the *UserProfileManager* via an *updateUserProfile()* method. The *UserProfileManager* then executes the update process and sends a message to the *RecommendationManager* for the successful update process.

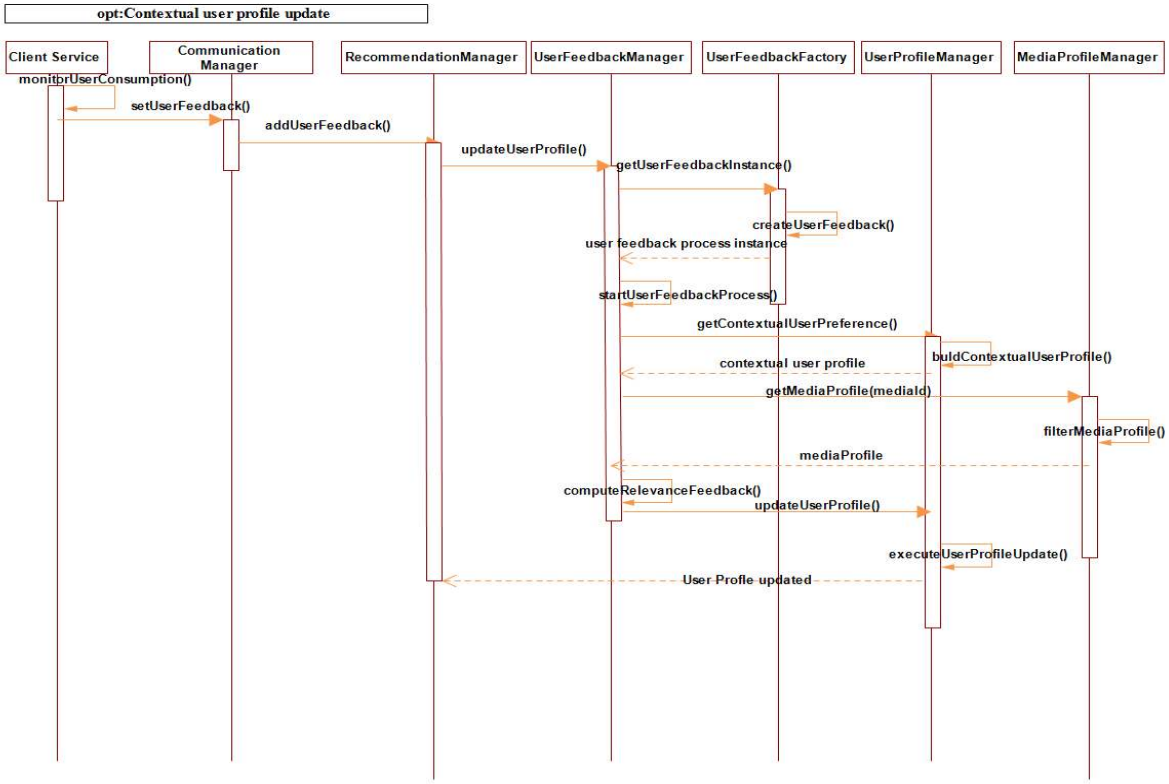


Figure 5.11- Simplified sequences of interactions for user feedback and contextual user profile update

5.7.3.5. Interactions for optional recommendation adaptation

The design allows interaction with external an adaptation service for optional adaptation of the recommended content according to contextual constraints, such as device characteristics and network conditions. However, adaptation processes and algorithms are out of the scope of this thesis, hence details of such implementation has been excluded. Nonetheless, the framework adopted the excellent multimedia adaptation engine and mechanisms developed by Andrade et al. in [127], [153]. The external adaptation service is invoked whenever the device, due to the constraints such as device's inability to support the display of the provided recommended content, and/or when network conditions such as bandwidths are inadequate. The external adaptation management service consists of Adaptation Decision Taking Engine (ADTE) and Adaption Engine (AE). The AE offers capability to perform a number of content adaptation operations, implementing, different encoding and transcoding mechanisms, whereas, the ADTE decides how and what kind of adaptation operations to be performed on the recommended content to suit the constraints that might be imposed by the user device and network conditions. Figure 5.12 presents the model of sequences of interactions that occur before and after adaptation process, please note that this is a simplified version of the adaptation sequence, which starts when the *RecommendationManager* receives a message, *adaptSelectedItem()* containing information about the selected item, from the client service about the constraints preventing the successful display of the recommended content. The *RecommendationManager* then invokes the *AdaptationManagement* service to initiate the adaptation process, which follows the sequences shown in Figure 5.12.

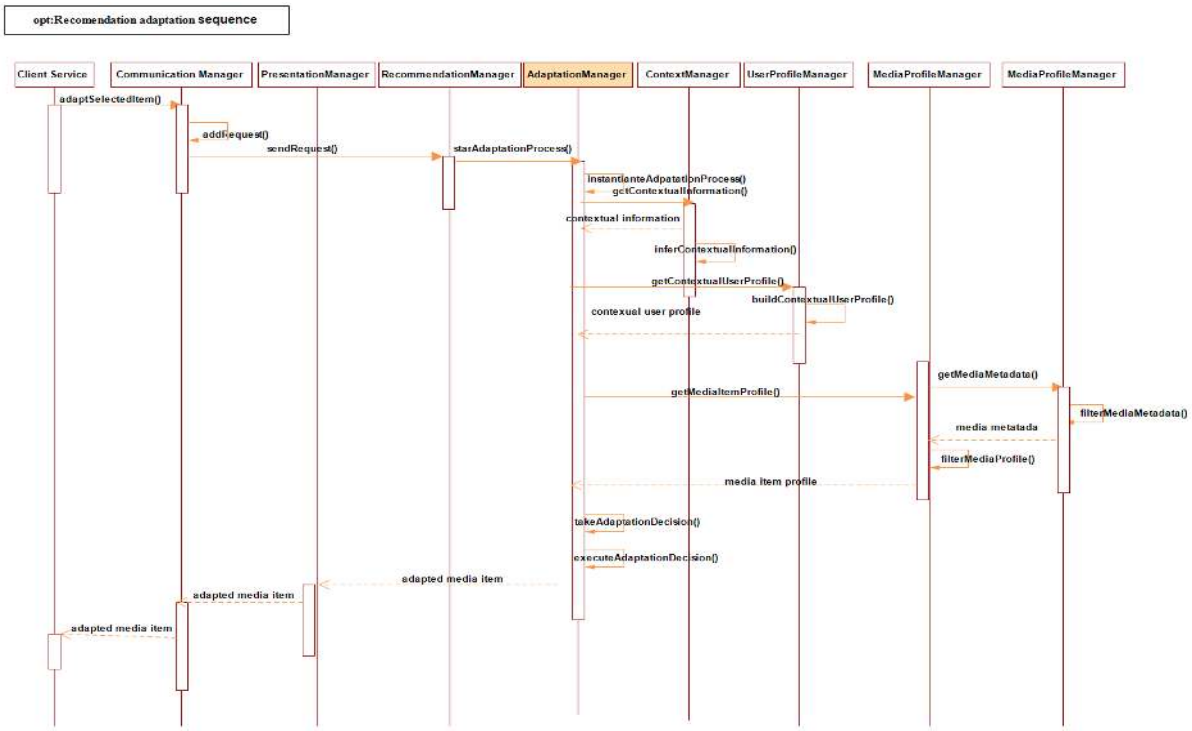


Figure 5.12- Sequence of interactions during recommended content adaptation

5.7.4 Client subsystem implementation

This subsystem as illustrated in Figure 5.13 comprises modules that are responsible for the dynamic recognition of user's consumption contexts as well as for obtaining user's initial basic profile and preference information (the first time they use the application). Additionally, it consists of the recommendation client module, which implements the recommendation client service. In this layer, we have the context monitors, context aggregators, context classifiers, context sensors, context recognizer, and basic user profile modules. The first five modules together constitute the context provider service.

The activities of these modules are depicted in Figure 5.14, showing the transition from sensing of raw sensor data to the classification of high-level context information, which could either be stored on the device or sent to the CAMR server subsystem. Additionally, the client subsystem contains modules that serve as recommendation client as well as those modules responsible for displaying media content. The recommendation service client is responsible for initiating and for obtaining recommendations, and displaying them to the users. Next, we describe the operations and functions of the modules of the client subsystem of the architecture, comprising the context and client service modules.

(A) Context sensing module

The context sensor module implements various sensing APIs of interest for accessing events that are produced by handheld device's sensors, preprocessing, filtering, extracting features, aggregating contexts, classifying, recognizing and preparing the context information for context consumers. This module consists of classes and packages that manage location sensors, network traffic and condition sensors, motion/activity sensors, weather sensors, environmental sensors, such as noise sensor, illumination monitor, device sensor and time sensor. Additionally, context monitor service interacts directly with the sensors modules to obtain raw sensor data at either predetermined intervals set by the user or on demand basis when user explicitly requests for recommendation.

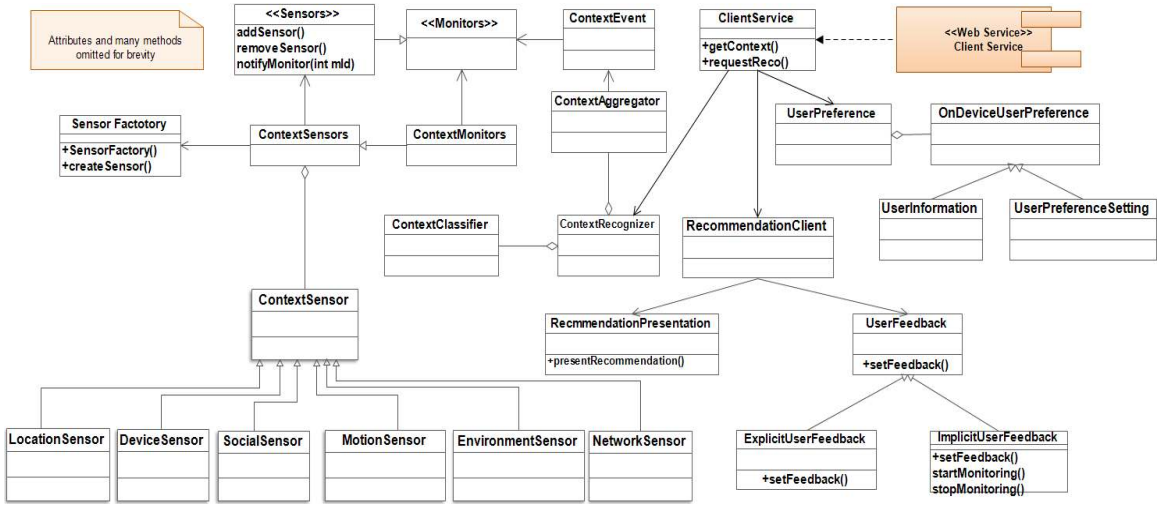


Figure 5.13- Analysis model and package references of the client subsystem of the proposed system

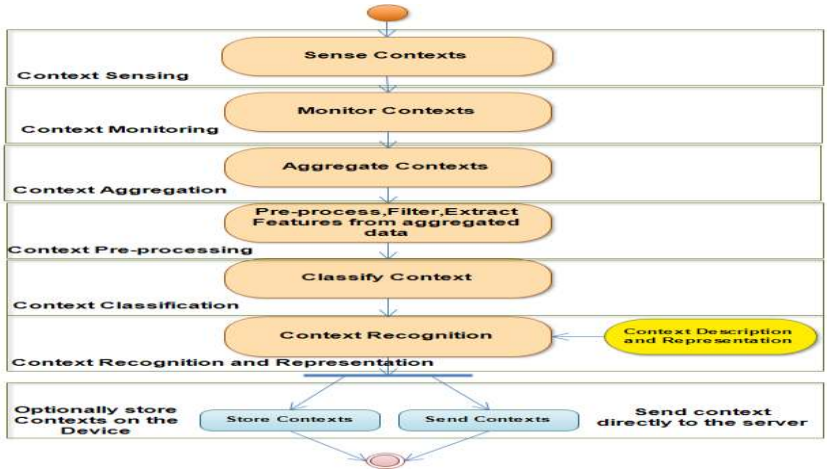


Figure 5.14- Activity model for context processing on the user device

Context aggregator is responsible for combining data from more than one sensor for providing more accurate and meaningful context information. For example, data from motion sensors such as accelerometers and gyroscope could be combined to improve the accuracy of activity context recognition. Additional responsibilities of this module include raw data pre-processing, filtering, feature extraction, etc. the processes, which have been described in the chapter three. The processed and filtered raw data, which have been extracted as features, are transformed by the classification class, producing high-level contexts from the extracted context features. The next module, context recognition module obtains the classified contexts from the classification class and then represents them in a format suitable for either persistence in device's database or direct communication to the server subsystem. In the next sections, the implementations of these modules are discussed.

(1) *LocationSensors*

The location sensors are those context sources from which low-level location information is obtained. There are two important location sensors implemented by this module. First, the GPS (Global Positioning System) sensor that

provides geographic location information. Information retrieved from this sensor is in the form of physical location longitude, latitude and altitude. Second, the location information could be obtained from Wi-Fi. Additionally, location information could be obtained from the CellID especially in situations where or GPS or Wi-Fi is not available.

(2) *MovementSensors*

The movement sensor package is responsible for sensing user activity contexts. It includes three important sensor types namely, the accelerometer sensor, the rotation sensor, and orientation sensor classes. The accelerometer sensor class implements APIs that senses acceleration of the device. The rotation sensor is responsible for sensing rotation information, by accessing synthetic rotation sensor, which calculates the rotation of the global coordinate system with respect to the device's coordinate system using accelerometer and gyroscope [112]. The rotation sensor represents the movement of the device as a combination of an angle θ and x, y, z-axes, in which the device has rotated. The orientation sensor is responsible for detecting the orientation of the device when held by the user. The movement obtains dynamic data about users based on their smartphone's orientation, rotation and acceleration.

(3) *EnvironmentSensor*

The environment sensor implements APIs for accessing information about the user's environment. Important sensor classes in this category are the light sensor, implemented as illumination sensor, the microphone, implemented as noise sensor. Another important environment sensor is the weather sensor which accesses information related to the environment temperature, humidity, weather forecasts, etc.

(4) *ContextEvent*

This interface contains *ContextEventListeners*, which propagate events about changes in contexts to interested or registered clients via its *contextChanged()* method in the form of notifications.

(5) *DeviceSensor*

DeviceSensor implements classes that obtain information about the characteristics, including capabilities, of the user devices. For example, information such as device screen size, refresh rates, current screen brightness, CPU usage, storage capacity and usage, supported coding formats, battery level remaining, etc.

(6) *NetworkTrafficAndConditionSensor*

Network traffic and condition sensor is responsible for sensing dynamic sensor traffics such as current and available bandwidths, maximum and minimum packet upload and download rates.

It also helps to determine the network interface, such as WiFi or cellular network interface to which the device is connected at the time of the recommendation.

(7) *ContextMonitors*

The *ContextMonitors* interface contains *addContextMonitor()* method, which can be implemented to add different processes that track changes of the events being produced by each sensor via their respective *ContextEventListeners*. Accordingly, each context sensor described previously has an equivalent context monitor, which periodically retrieves events produced by the respective sensor. Each registered context monitor is called periodically to acquire contextual information by calling its *getContextInformation()* method.

(8) *ContextAggregator*

The function provided by the *ContextAggregator* is the gathering of sensor events, especially events generated by sensors with similar attributes. For example, rotation, orientation and acceleration sensors all are tri-axial sensors and their events can be combined to improve context recognition quality and accuracy. Additionally, this module is responsible for sensor events pre-processing, such as noise reduction via filtering, as well as executing feature

extraction process, and conversion of the features into vectors, which serve as inputs to the subsequent modules, such as the context classifier and recognizer.

(9) *ContextClassifier*

ContextClassifier implements context classification model, which processes sensor data feature vectors to obtain labelled contextual information. Its functions consist in transforming the raw sensor data features into machine and human understandable high-level context information. Supervised classification models, such as kNN, kStar, Decision Trees, Neural Network, etc. could be implemented in this module to realize context classification.

(10) *ContextRecognizer*

Context Recognizer is the coordinator of all other context modules, such as *ContextMonitors, ContextSensors*, *ContextClassifier*, and *ContextAggregator*, running on the user device. Via *ContextMonitors' ContextEvent*, it determines what to do whenever any significant change in the user context occurs and triggers communication between the client and the server subsystems, especially initiating the recommendation process. In addition, it could use periodic probing of context monitors to keep track of the contextual changes. If there is a significant change in the user's contextual information, it communicates its decision to the server in order to trigger the recommendation process via the *RecommendationManager*. The *ContextRecognizer* uses two main classes, *ContextAggregator* and *ContextClassifier*, to identify dynamically, user's contexts. In turn, *ContextAggregator* collects events via *ContextEvent* class from different sensors, aggregating them for pre-processing, and feature extraction processes. *ContextMonitor* class is responsible for detecting changes in context events via event listeners for different sensors, before the *ContextAggregator* collects those events. The *ContextSensors* implement APIs that directly access the hardware sensors on the device to obtain raw data. As stated above, the raw sensor data are processed by the *ContextRecognizer* and its helping classes, especially *ContextClassifer*, which implements classification algorithms to obtain high-level context features. The contextual features, obtained from the *ContextRecognizer*, are presented as resources represented either in pure XML representation or in other XML-based descriptions, such as MPEG-21 UEDs and can be accessed via public Web service interfaces. The context information resources could be consumed by external sources apart from those in the server subsystem of the proposed system. One other important function of the *ContextRecognizer* is that it is also responsible for storing important context information on the device. For example, users can specify location preferences, in the form of Point of Interest (POI), which is used by the device for implicit recognition of the user's locations. The functionality and resources provided by the *ContextRecognizer via the ClientService* are exposed as a Web service interface, where context resources could be pulled (queried) or pushed to the consumers.

(11) *SensorFactory*

Finally, in the sensory module we have the *SensorFactory* class. This class is responsible for creating and instantiating sensors. Its major function is to create sensor objects, which are specified in the *ContextSensor* class, using its *createSensor()* method. With this class, sensor can be created and instantiated when needed.

(B) UserPreference

The UserPreference class manages the on-device preferences of the users. These preferences, which may include sensitive user information, are those managed by the users themselves on their devices. For example, there are some basic POI (point of interest) location information that is set by the users. These POIs help the system to know exactly the choice locations of the users so that when the *ContextRecognizer* discovers location that matches any of the POIs, the system can use this information in addition to the activity context as well as other contextual information to trigger implicit contextual recommendation process. *OnDevicePreferenceSetting*, which is extended by the *UserPreferenceSetting* and *UserInformation* classes assist the *UserPreference* class on the mobile client side to manage this type of information. This information at user's consent are sent to the server side for contextual user profiling and recommendation processes.

(C) *RecommendationClient*

The *RecommendationClient's* primary functions is to initiate recommendation process and to display the recommended items to users as well helping to collect explicit or implicit user feedback. The *UserFeedback* class is extended by the *ExplicitUserFeedback* and *ImpliciUserFeedback.* The *ExplicitUserFeedback* class monitors user's interaction with the system and collects observations such as specific types of content users usually consume and in what contexts.

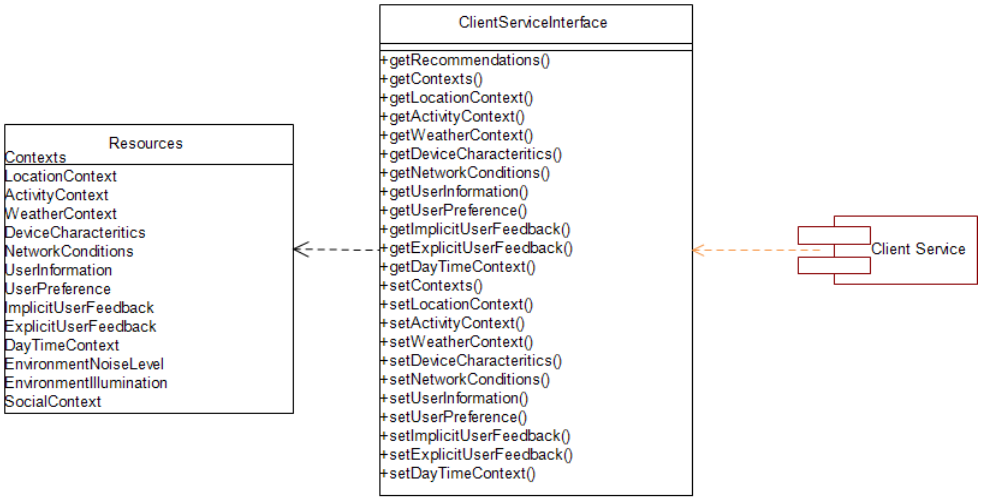


Figure 5.15- Client service interface showing resources that are exposed by the web service

For example, it could determine that the user always consumes music content when driving or watching movies during dry weekends. On the other hand, the *ExplicitUserFeedback* directly asks users to rate items they have consumed and collect the rating to be submitted to the server side's *UserFeedback* module.

(D) *ClientService*

The *ClientService,* illustrated in Figure 5.15, is the module responsible for the coordination of services provided by the client subsystem, especially the recommendation presentation, basic user preference management, user feedback acquisition, user information, etc. via *RecommendationClient* class. It is assisted by various classes to realize these functions, for example, the *RecommendationPresentaton* class helps to display and present recommendation list to users. The *RecommendationPresentation* class also provides means by which recommendations can be provided to users either by explicit request (e.g. by pushing a button) or implicitly according to user's context changes. Additionally, via *ReccommendationPresentation* and *UserFeedback* classes it provides services that display where users can optionally provide explicit feedbacks on the recommended items. The *UserPreference* class in this module manages some optional preferences that users are allowed to set on their devices. For example, user's point of interest (POI) or preferred presentation formats could be set by the users and persisted on the device, whereas the module provides a class, which manages user's information such as names, sex, age, profession, etc., which is persisted on the user device. There are two reasons for this. One, because users are always very reluctant to release personal information to third party systems, to guarantee user's trust in the system, these provisions are made to keep sensitive information on their devices. Second, it is meant to relieve the server of some basic processing overheads, which could be done comfortably on the user's device. This second factor explains why context service is implemented on the device. The *UserPreference* class in this module manages some optional preferences that users are allowed to set on their devices. For example, user point of interest (POI) or preferred presentation formats could be set by the users and persisted on the device, whereas the module provides a class, which manages user's information such as names, sex, age, profession, etc., which is persisted on the user device. There are two reasons for this. First, because users are

always skeptical about releasing personal information to third party systems, to guarantee user's trust in the system, these provisions are made to keep sensitive information on their devices. Second, we want to relieve the server of some basic processing overheads, which could be done comfortably on the user's device. This second factor explains why context service is implemented on the device. More importantly, it exposes an interface as a web service.
In practice, the *ClientService* interface is exposed as either REST or SOAP based Web service (our present implementation is based on REST Web service) transmitting recommendation request to the server subsystem as well as providing contextual information as Web service resources that could be consumed by external services such as the server side services (e.g. for user profiling, personalization and recommendation purposes). For example, the current implementation, based on REST Web service, obtains the user's current location, via an HTTP GET message is sent via the communication management service to the client subsystem and a response containing the location information to the CAMR server.

The client service is created with references to various resources that it provides for the consumers/ clients/requestors via the unified HTTP interface. Let us take REST based implementation of the service as an example. REST is an architectural style for distributed hypermedia system. A RESTful web service is an API that is accessed through the HyperText Transfer Protocol (HTTP), which is executed on the remote system that hosts the services being requested. REST service is implemented, using HTTP based on the principles of the REST architectural style. The exposed resources are shown in the left side of Figure 5.15 with caption "Resources". Each resource as shown in Table 5.23 defines operations of the REST API by annotating methods with REST annotations, i.e. @GET, @POST, @PUT, and @DELETE. The @GET operation is used to transfer the representation of the requested resources, e.g. contextual information of a particular user, from the provider (the client service) to the requestor (the server subsystem in our case). This operation would be useful in situations where server specifically makes a request to the client for the resources it provides. The @POST operation is used to post the representation of the affected resource via a handler to the server. The requestor then validates and verifies the identity of source of the POSTed information. This operation can be performed to update existing resources for example, which is located on the server.

```
@POST
@Path("/usercontexts/{userId}")
@Consumes({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
public Response postUserContext(JAXBElement<UserContext> highLevel,
                @PathParam("userId") Integer userId) {
        User user = userData.getUser(userId);
        if (user != null) {
                UserContext userContext = userContext.getValue();
    URI highLevelUri =
uriInfo.getAbsolutePathBuilder().path(highLevelContext.getHighlevelId().toString()).build();
          return Response.created(userContextUri).build();
        }
        return null;
}
```

Figure 5.16- POST request for user's contextual information

```
@GET
@Path("/usercontexts/{userId}")
@Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
public List<UserContext> getUserContexts(@PathParam("userId") Integer userId) {
        userContexts = retrieveAllContexts(userId);
        if (networkList.get(0).getDevice().getUser().getUserId() != userId)
                return null;
        return userContexts;
}
```

Figure 5.17- GET request for user's contextual information

Table 5.23 - Examples of REST implementation of client service resources

Resource	*REST URI*	*Explanation*
Contexts	*context/usercontexts/{userId}*	*All contexts of a user with id userId*
LocationContext	*/contexts/location/{userId}*	*Location context of a user with id userId*
ActivityContext	*/contexts/activity/{userId}*	*Activity of a user with id userId*
WeatherContext	*/contexts/weather/{userId}*	*Weather of the location where user with id userId is currently present*
DeviceCharacteritics	*/contexts/device/{userId}*	*Characteristics of the device of user with id userId*
NetworkConditions	*/contexts/network/{userId}*	*Nework conditions of the network to which the device of user with id userId is currently connected*
UserInformation	*/user/userInfo/{userId}*	*Basic information of user with id userId*
UserPreference	*/user/userpref/{userId}*	*Preference settings of user with id userId*
ImplicitUserFeedback	*/feedback/implicit/{userId}*	*Implicit feedback from user with id userId*
ExplicitUserFeedback	*/feedback/explicit/{userId}*	*Explicit feedback provided by user with id userId*
DayTimeContext	*/contexts/daytime/{userId}*	*Day and time context of the location where user with id userId is presently located.*
EnvironmentNoiseLevel	*/contexts/noise/{userId}*	*Noise level of the location where user with id userId is currently located*
EnvironmentIllumination	*/contexts/illumination/{userId}*	*The illumination of the location where user with id userId is currently located*
SocialContext	*/contexts/social/{userId}*	*Social contexts e.g. whether the user is currently logged in to Facebook, twitter etc. of user with id userId*

The @PUT operation is used to update an existing resource but unlike the update operation of the @POST, it overwrites the existing resource being updated. @DELETE operation can be issued to delete resources on the server. For example, a client can issue this operation to delete resources that are sent to the server accidentally, provided such client is authorized to do so. Figures 5.16 and 5.17 illustrate examples of Java snippets of POST and GET request operations by the server on user's contextual information resource from the client service. Please, note that contextual information contains all the contexts of the user at the time of request.

5.7.5 Server subsystem design

We describe, in this section, the design of different aspects of the functional architecture of CAMR server subsystem. Figure 5.18 illustrates the high-level functional architecture, showing its main modules and their interactions. In Figure 5.19, we give a more detailed view of the implementation modules of CAMR server. All modules interact via interfaces, please note however that Adaptation Management is not directly implemented as part of the CAMR server subsystem, as we have explained before. *Recommendation Management* is the central module of the framework, establishing direct or indirect connections between the other modules. It provides the service that executes tasks that generate recommendations. The *Media Profile Management* connects online-based media content service providers, such as YouTube, IMDB, etc. to extract media metadata descriptors. The *Context Management* coordinates processing of contextual information received from CAMR client subsystem and instantiates it into the *Context Inference Management*. The *Context Inference Management* is responsible for relating two or more recognized context information to derive semantically expressive high-level context information, using ontology and SWRL model and other processes described in chapter three. The *Contextual User Profile Management* collaborates with *Context Inference Management* and *Context Management*, taking the context information, user preferences, and establishing a relation between the context information and user preferences.

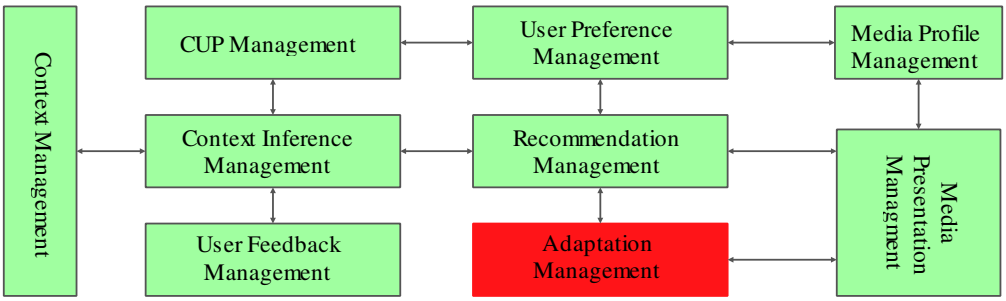


Figure 5.18- High-level architecture of CAMR server subsystem illustrating its functional components and their interactions

The *User feedback management* is responsible for learning the user's interaction with the framework; it collects either explicitly or implicitly, the user's contextual feedback in order to improve the quality future recommendations, which truly reflects the user's contextual preferences. *The user preference management* is responsible for scoring the contextual preferences of the user, and for tracking those media items that users have consumed before, which might no longer be interesting to them. The *Media presentation management* is responsible for determining the appropriate way to display recommended media items by the mobile phone. The *adaptation management* is responsible for determining if the presentation format of the recommended media items can be played by the user's device. If the device cannot play the content, it then determines the appropriate adaptation mechanism to execute in order to display the media item. Each of these modules of CAMR server subsystem will be described in more detail in the next subsections.

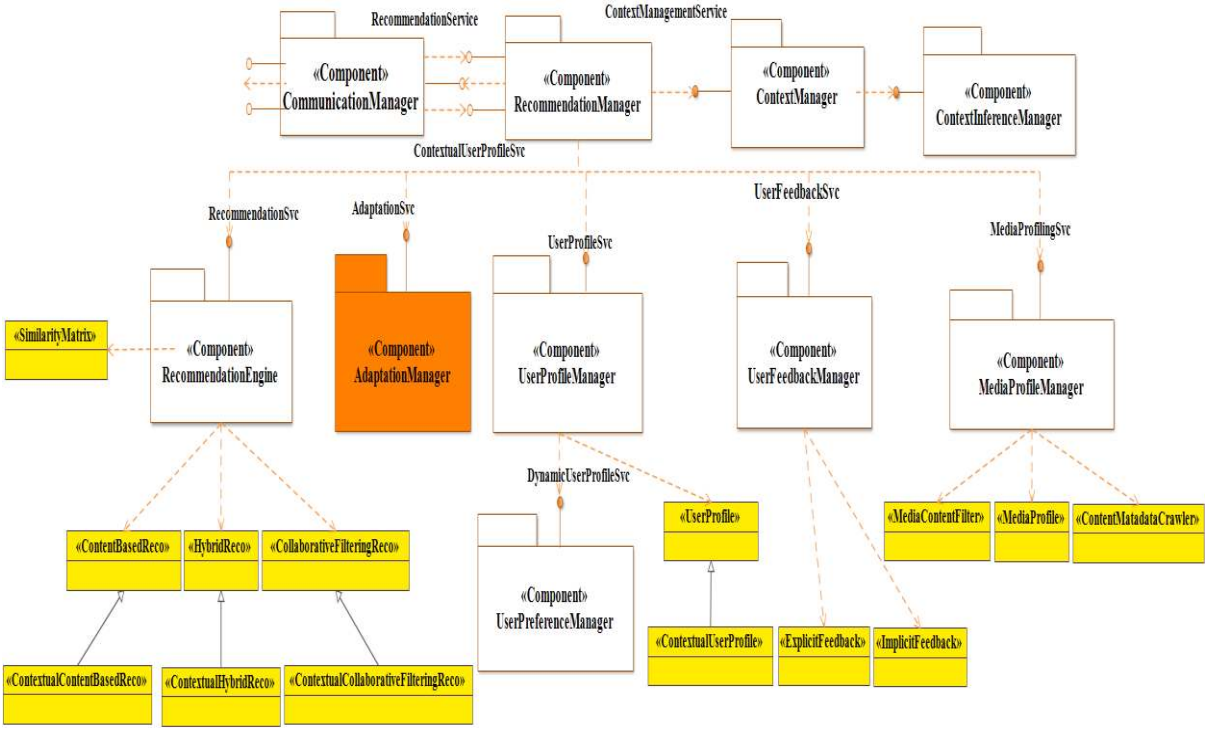


Figure 5.19- Detailed UML model of CAMR server subsystem of the proposed framework

5.7.5.1. *RecommendationManager* package

RecommendtionManager is the central module of the system, establishing direct or indirect connections between the other modules and tying together their functionality, assisted by other modules to realize the overall objective of the framework. It provides coordination of services that execute the tasks that generate contextual recommendations. Essentially, *RecommendationManager* is assisted by *RecommendationEngine* class to coordinate activities of the recommendation processes. In addition to the *RecommendationEngine,* this package contains seven other principal classes, which implement contextual content-based recommendations, contextual collaborative filtering based recommendations and contextual hybrid recommendations, with the capability to execute non-contextual or traditional recommendation processes, as depicted in Figure 5.20. This design achieves one important goal. It provides more specialized recommendation services, i.e. contextual recommendation processes, and yet it enables the re-use of the traditional recommendation processes.

When *RecommendationManager* receives message about user's *context* changes from the client subsystem's *ContextRecognizer's setContextMethod()* via *CommunicationManager*, or when a user makes an explicit recommendation request, it sends the message to the *UserProfileManager* asking for the contextual profile of the user. The *UserProfileManager* then retrieves the contextual information it receives from the client subsystem and sends it to the *ContextManager*, which then parses and instantiates the contextual information in the inference model assisted by the *ContextInferenceManager*. The *ContextInferenceManager* then processes the contextual information that contains context features, to derive high-level contextual information, after receiving the high-level context information from the *ContextInferenceManager.*

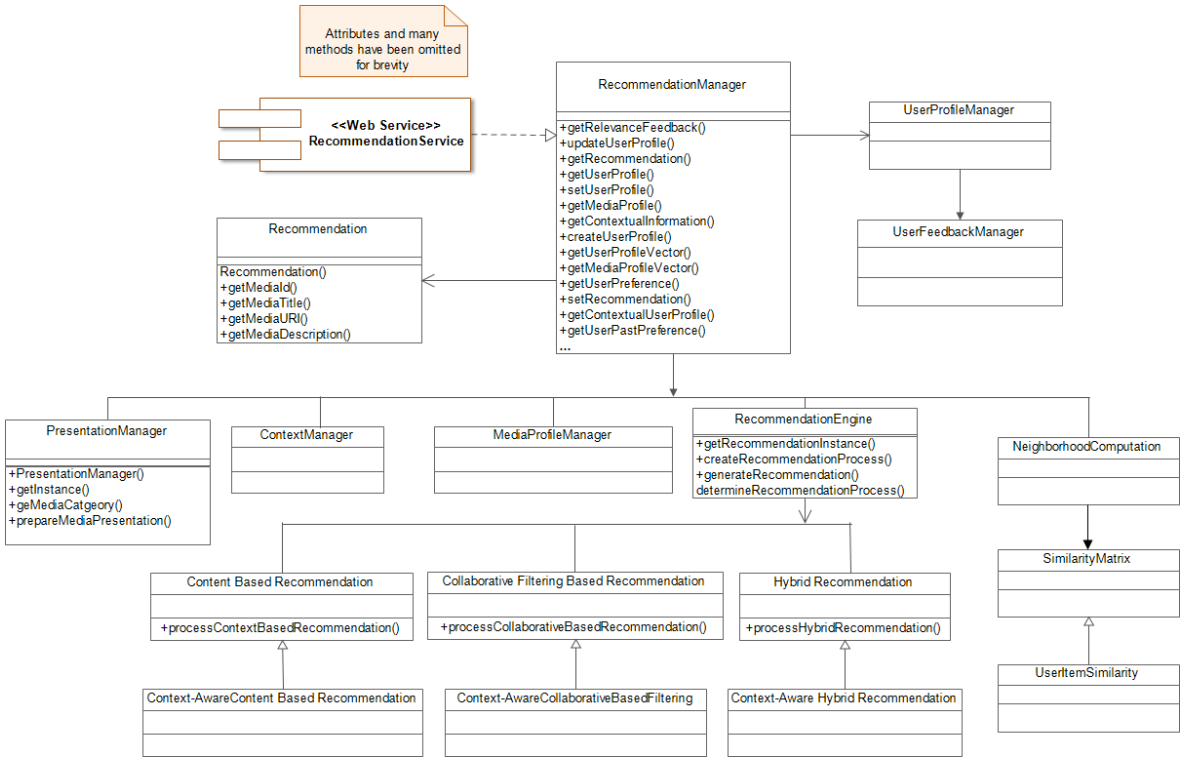


Figure 5.20- Excerpt from the recommendation manager class model showing its most important classes

Additionally, based on the type of user profile (i.e. whether it is contextual or not) the *RecommendaionManager* can switch between contextual and non-contextual recommendation processes.

The *RecommendationEngine* also coordinates user profile update and relevance feedback processes via its *updateUserProfile()* method. The *RecommendationEngine* requires and implements four major interfaces. These include *ContentBasedReco*, which exposes all operations required to realize content-based recommendations. The *CollaborativeFilteringReco* exposes the operations needed to implement collaborative filtering based recommendations. The *HybridReco* exposes the operations required to implement a combination of collaborative and content-based recommendations. The contextual recommendation interfaces extend their respective recommendation interfaces, the processes used by any class implementing these interfaces have been discussed in chapter four. The fourth interface is the *SimilarityMatrix*, which is an interface that is used to represent and manipulate the pairwise similarity scores between items being processed for recommendations, where in this case items could be users and candidate media items. The pairwise similarity scores could be similarity between candidate media items, between users, or between users and candidate media items depending on the type of recommendations being computed. The *RecommendationManager* exposes two interfaces as Web services. The *ContextualUserProfileService* and the *ContextualRecommendation* services. The *ContextualUserProfileService* exposes all operations related to contextual user processes notably *getContextualUserProfile(), getUserProfile(), buildUserProfile(), inferUserPreference(),* etc. These operations are explained further in the *UserProfileManagement* package section.

To external sources, it exposes a *RecommendationManager* service, which for example, services such as the client service can interact. Through this service interface, client service can send *getRecommendation()* message with the user's current contextual information as well as user preference information as parameters, and in return receives contextual recommendations.

5.7.5.2 *UserProfileManager* package

Whenever *RecommendationManager* receives the high-level contextual information from the *Client Service via the CommunicationManager,* it sends this information to the *ContextManager*. The *ContextManager* is responsible for providing a more meaningful and higher-level contextual information from the contexts obtained from the client device, assisted by the *ContextInferenceManager*. The *UserProfileManager* takes the received contextual information from the *ContextManager* to build a contextual user profile. The contextual user profile adapts the user profile from which the current user's preferences can be inferred based on the context in which he has preferred certain media items in the past. Additionally, it has the capability to generate profile information for a new user who is using the system for the first time. Specifically, three main classes are provided in this package as illustrated in Figure 5.21.The *UserProfileManager* class, the *UserProfile* class, *ContextualUserProfile*, and the *ContextualUserProfile* class. *UserProfile* is the generalized user profile class, which is responsible for context-independent user profiling, it helps the *UserProfileManager* to manage the traditional user profile.

It allows recommendations for users in some instances such as when user's contextual information is not available for one reason or another. The *ContextualUserProfile* is a specialized *UserProfile* class that helps the *UserProfileManager* to build and process the contextual user profiles using contextual user information and preferences for contextual recommendations. The *UserProfileManager* is also assisted by the *UserPreferenceManager*, which processes and infers user preferences according to their contextual situations. Additionally, the *UserProfileManager* is designed with methods for creating user profiles (*createUserProfile)* assisted by the *UserProfileFactory* class, updating user profiles(*updateUserProfile)*, for retrieving user profiles(*retrieveUserProfile),* for deleting user profiles(*deleteUserProfile)*. The *UserPreferenceManager* class also is responsible for the management of the dynamic user preference, generating the numerical representation of the user's dynamic preference using two parameters. The first parameter is the lifetime of the user's contextual preference, which keeps track of the consumption preferences of the user.

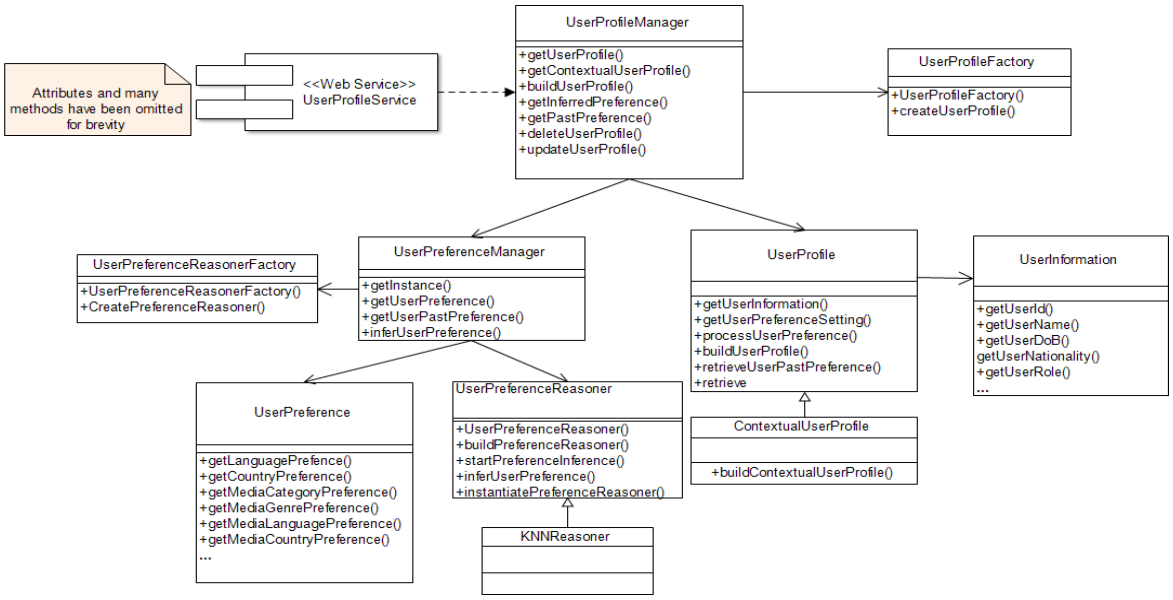


Figure 5.21- Simplified class model of user profile manager showing its relationship with contextual and non-context user profile classes.

The lifetime is the time elapsed since the user last consumed recommended items belonging to a category-genre-property in a given context as discussed in chapter four. The second parameter is the weight, which represents the relevance of user preference in context. In addition, these parameters are used in the user feedback profile update process.

In order to learn or infer user preference in the current context as stated earlier, the *UserProfileManager* uses the *UserPreferenceManager* working with a *UserProfileReasoner* class, which implements reasoning or learning mechanism, such as kNN based algorithm, e.g. case based reasoning, to infer user's dynamic preference in the current context using its *inferUserPreference()* method. This method implements user profile and preference process described in section 4.4.4 of chapter 4. Additionally, the *updateUserProfile()* method implements the update process described in section 4.4.7 of chapter 4, adopting both explicit and implicit relevance feedback mechanisms. The *UserProfileManager* requires the *UserProfile* interface, which contains the representation of a user profile as well as operations being performed on the profile. *ContextualUserProfile* extends the *UserProfileManager* exposes its important operations such as *buildUserProfile(), getUserProfile(), inferUserPreference(), as UserProfileService*. These methods implement processes and algorithms that are required to process users' profiles, based on their contextual information as discussed in the contextual user profile model in chapter four.

5.7.5.3 *MediaProfileManager* package

This package contains classes such as *MediaProfileFiltering* and *MediaProfileFactory* classes as depicted in Figure 5.22. The *MediaProfileFiltering* helps the *MediaProfileManager* to obtain candidate media items metadata via *MediaMetadataCrawler* and *MediaMetadataExtractor*, which implement processes that crawl and parse respectively, online-based media items' metadata, from external online media sources. The class *MediaMetadataExtractor* is designed for parsing pure XML or XML-based media descriptors such as MPEG-7 media metadata descriptors as well as other descriptors such as JSON.

In addition, the *MediaMetadataCrawler* class implements and manages publicly available APIs for obtaining metadata of candidate media items from online-based databases. For instance, it could connect with internet-based databases

such as IMDB, Last.fm, etc. to obtain metadata of candidate movie and music items respectively. The *MediaProfileFatcory* class helps the *MediaProfileManager* to create a profile for each candidate media item, from which a media profile vector is computed with the help of the *MediaProfileFilteringManager*, using *MediaProfielFiltering* and the *MediaMatadata* classes to filter and compute the media vectors for all candidate media items respectively. Additionally, it contains the *MediaProfile* class, which works in tandem with the *UserProfileManager* to manage information contained in each candidate media item profile.
The *MediaProfile* class is also used to manipulate media metadata stored in the media profile repository. More importantly, the *MediaProfileManager* coordinates the activities of all these classes to create, to delete, to update and to manipulate media item profiles via *MediaProfile* class. Please note that the *MediaProfile* class extends the *MediaItemCategory* class, which distinguishes between categories to which each candidate media item belongs. The *MediaProfileManager* exposes *MediaProfilingService* interface, which provides necessary operations of the package to other modules.

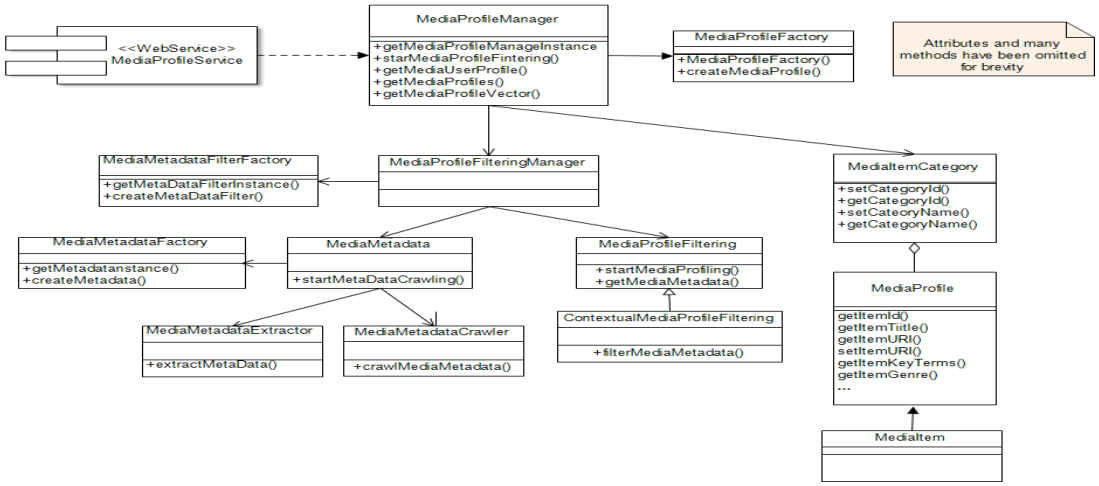


Figure 5.22- Media profile manager simplified class model

5.7.5.4. *ContextManagement* Package

This is one of the most important packages of the proposed framework. The package as illustrated in Figure 5.23, contains all classes that are responsible for the contextual information processing. The context information is obtained from the client subsystem, which means that the *ContextManagement* package works in tandem with the *ContetRecogizer* package of the client subsystem. The *ContextManager's* primary function is to coordinate the activities of its constituent classes. These classes represent various categories of contextual information. Intuitively, it contains three main classes. These classes are the *ContextExtractor*, *ContextualInformation* and *ContextInferenceManager*. The *ContextualInformation* class represents and manipulates different kinds of contextual information, such as user activity, user location, time and day contexts, environment contexts, such as illumination and noise level as well as weather information. Others are the device characteristics, network characteristics and conditions, social contexts, etc. Additionally, the *ContextExtractor* class implements methods for validating, parsing, and extracting contextual information. For example, there are various methods implemented by this class for parsing XML based MPEG-21 UED representation of the each contexts. Finally, this package also contains *ContextManager* class, which is responsible for creating and instantiating the entire context objects and for manipulating and coordinating their activities.

The *EnvironmentContext* represents and processes contextual information related to the media consumption environment, such as illumination and noise level. As depicted in Figure 5.23, *EnvironmentContext* aggregates *WeatherContext*, which implements APIs and Web services for processing environmental weather information such

as weather forecast, maximum and minimum temperature, etc. The *EnvironmentContext*, in addition, processes contextual information such as environment illumination and noise level. In addition, the *ContextManager uses the ContextInferenceManager to obtain* high-level context information via the *ContextInference* class. Note that for brevity, *EnvironmentContext*, *WeatherContext* and other context types are not shown in Figure 5.23, but they all extend the *Context* class.

5.7.5.5 *ContextInferenceManagement*

The *ContextInferenceManager* shown in Figure 5.24 is a module within the context management module providing support for relating two or more recognized context information to derive semantically expressive higher-level context information. This package contains all classes that implement the context ontologies and rule based inference, described in chapter 3. There are three most important classes contained in this package. First is the *OntologyManager* class, which coordinates the ontology and inference processing. The *OntologyManager* uses two important methods,

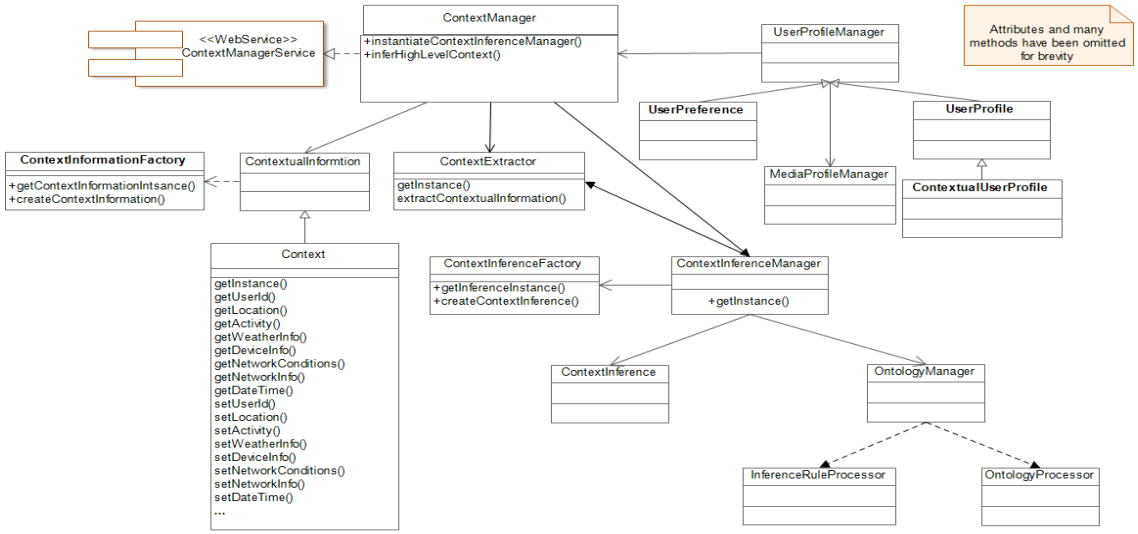


Figure 5.23- Excerpt of context manager class model

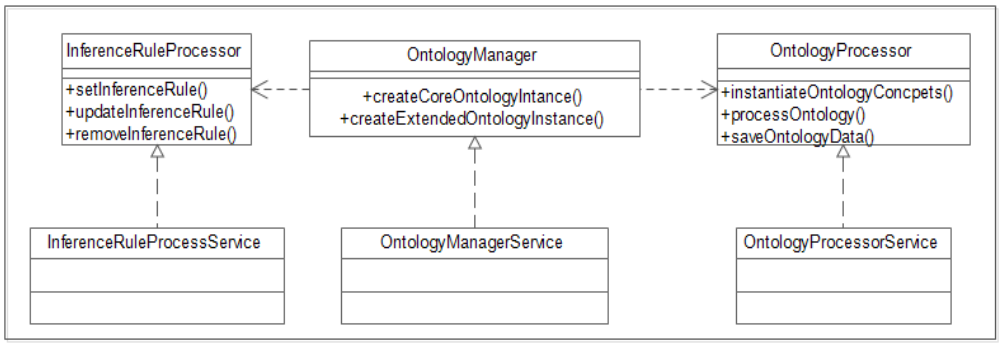


Figure 5.24 - Context inference manager model based ontological reasoning and inference

createCoreOntology() and createExtendedOntology(). The *createCoreOntology()* method is used to create user's context ontology independent of specific application scenario. On the other hand, *createExtendedOntology()* method is used to create additional context concepts as well as rules that relate these context concepts in order to realize higher-level level contextual information from those obtained from the client service. *OntologyProcessor* and

InferenceRuleProcessor classes are responsible for the manipulation and processing ontology concepts and classes, as well semantic rules, which relate one context concept to another to obtain more meaningful contextual information.

Note that the original contextual information is obtained from client service and then instantiated by the *ContextManagement* service into the *OntologyManager* for inference purposes.

5.7.5.6 *UserFeedbackManagement*

The *UserFeedbackManager* illustrated in Figure 5.25 contains classes that implement learning of user's feedback and interactions with recommendations. It contains two mains classes that help to provide information about the interactions of users with the provided recommendations in order to improve the quality of future recommendations. It collects, either explicitly or implicitly, user's contextual feedbacks in order to improve the quality of future recommendations, which truly reflects a user's contextual preferences. This module works with the *UserProfileManager* to update the user's contextual profile after the user has been provided with recommendations. Irrespective of whether the user consumes the recommendation or not, the feedback reports user's interaction, providing necessary information for updating the user's contextual profile to reflect user long and short-term preferences. For example, two important classes are contained in this module. The *ExplicitUserFeedback* and *ImplicitUserFeedback* classes, which extend the *UserFeedback* class. The former is responsible for collecting user's satisfaction information via the recommendation presentation interface provided on the user's device, by asking users to provide numerical values of their satisfaction/ relevance of the recommended items in the respective contextual situation. The latter is responsible for collecting information using user agent that monitors the interactions of the users with the recommendations.

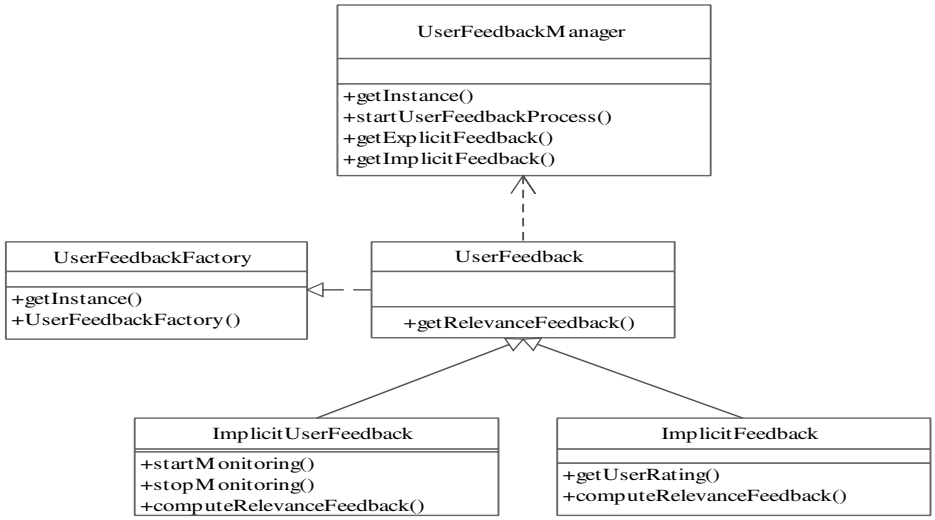


Figure 5.25 - *FeedbackManager's* simplified class model

5.7.5.7 *MediaPresentationManager* Package

The *MediaPresentationManager* is responsible for the preparation of the recommended media items for display on the mobile client. This package contains classes that determine the appropriate display options for the item among the recommended items selected by the user. For example, if the selected item is a movie, it displays the movies with more appealing information such as posters, backdrop, synopsis, trailers, or providing links to watch the full movies provided the user has the authorization to watch the movies via that link. This module works in tandem with the device's user interface module via a Web service interface, *MediaPresentationManagementService,* which exposes its operations to the user interface module of the client subsystem as well as to *AdaptationManagement* and

RecommendationManagement modules. Its main function is to determine the best possible way to display the recommendation presented to users.

5.7.5.8 *AdaptationManager*

The *AdaptationManager* is responsible for managing external adaptation service, which determines if the presentation format of the recommended media items can be played by the mobile device or not. If the device cannot play it, it then determines the appropriate adaptation mechanism to be executed in order to display the media item. This package consists of classes that extend the *Adaptation Decision Engine* (ADE) and *Adaptation Engine* (AE) suites developed by [153]. The ADE based service should offer capabilities to perform optional adaptation operations on the recommended item, especially the audiovisual content. Such adaptation operations include, among others, scalable, non-scalable, and summarization operations. The ADE based classes receive necessary contextual information, as well as user preferences, encoded in MPEG-21 UEDs, from the *ContextManager*, and instantiates it into the context inference engine in order to decide via its Context*InferenceManager module* on whether adaptation operation is required. If adaptation is required, it determines the kind of adaptation operation to perform. Its final decision is sent to the adaptation engine that performs the requested adaptation operation on the recommended content. Let us assume that the system discovers low battery problem during recommendation process, i.e. when user has selected a recommended multimedia content. One possible adaptation to perform is a summarization of the video content. For example, the typical context information obtained by the adaptation decision engine (ADE) is given in Table 5.24. The information related to the movie's visual data is given in Table 5.24. From these tables, it is evident that some form of adaptation is required to provide acceptable quality that satisfies those constraints in Tables 5.24 and 5.25 for successful display of the movie clip.

Table 5.24 - Usage environment context information

Terminal Context	Maximum Frame Rate	25fps
	Battery Time Left	2 minutes
	Display Size	WVGA(480 x 800)
Network Context	Available Bandwidth	142 kbps
Environment Context	Noise Level	85.5 dB
User Preference	Frame Rate	30fps
	Frame Size	Xhdpi(720 x 1280)

Table 5.25 – Content information

Media Visual Coding	Frame Size	Xhdpi(720 x 1280)
	Frame Rate	30fps
	Bit Rate	1024 kbps

Table 5.26 – Example of adaptation decision parameters

	Input Values	Output Values
Frame Rate	30fps	15fps
Bit Rate	1024 kbps	142kbps
Spatial Resolution	720 x 1280	480 x 800
Volume Level	0.9	N/A

Table 5.26 illustrates an example adaptation decision parameters sent to the adaptation engine by the adaptation decision engine to adapt the selected recommended movie item in order to minimize power consumption thus, allowing the display of the content. Since the implementations of both adaptation decision engine and adaptation engine are beyond the scope of the thesis, details of its design and implementation are, therefore, not discussed further. However, readers interested can consult [127], [143], [153] for more details.

5.7.5.9 *CommunicationManager* Package

The *CommunicationManager* functions like a broker or a connector between communicating entities of the architecture, managing in particular the communications between the client subsystem and the server subsystem. Additionally, it handles communications between modules of each subsystem. It offers two different modes of operations, which are selected depending on whether the request is explicit or implicit. As illustrated in Figure 5.26, it provides three major functions. First, it is responsible for the discovery of services via service addresses such as service URIs, provided by CAMR framework. In this function, it establishes connections between the client subsystem and the server subsystem, for example, and captures all connection errors and exceptions during communications. After communication between CAMR entities has ended, the communication manager then disconnects the communicating entities and releases the resources, especially the client side resources. The second function of the communication manager is request dispatching. As soon as connection between communicating entities is established, the communication manager then sends the requests on behalf of the requestor (client), to the calling service, serializing the requests into streams of bytes to be transmitted over the network, using HTTP. Once the request has been dispatched, the communication manager then can either block and return the response to the client, or provide an asynchronous response handler, which enables the client to focus on other tasks, while awaiting a response from the service.

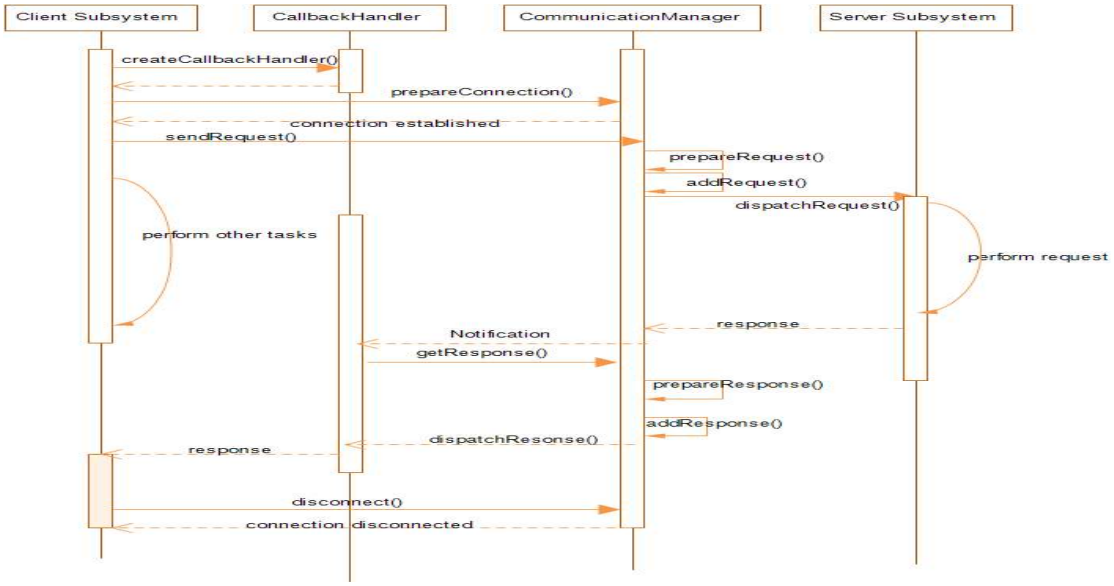


Figure 5.26 - Illustrates the interactions between client and the server mediated by the *CommnicationManager*

For example, the client subsystem uses the blocking mode in situations where users make explicit recommendation requests. On the other hand, if the request were implicit such as when the client based on contextual changes decides to make a recommendation request, the asynchronous response handler with callback would be used. The third function of the communication manager is the reception of the service response on behalf of the client. It provides the functionality that helps client to deserialize the streams sent from the service into a data format that the client can understand. The main methods of the *CommunicationManager* are described in Table 5.27. Thus, based on the function defined above, the *CommunicationManager* handles the requests from the client subsystem. In particular, it illustrates the exchange of messages when recommendation request originates from the client side.
The client subsystem first creates a callback handler and then sends a request (e.g. the recommendation request), including a reference to the callback handler to the *CommunicationManager*, which then starts a new thread that dispatches the recommendation request to the *RecommendationManager*.

Table 5.27 - Communication Manager main methods and their descriptions

Method	Function
createCallBackHandler	The callback handler is responsible for handling events such as when recommendation request is made either based on context changes or on explicit user request. It allows the client side to continue with other task while the request sent to the CAMR server is being processed. It collects the results (list of recommendations when it is ready and contacts the client to deliver recommendation list.
prepareConnection	This method as it name suggests prepares the connection between the client side and the server side.
sendRequest	This method is responsible for sending the request from the client to the communication manager.
despatchRequest	Communication manager uses this method to deliver the request it receives from the client to the server
prepareResponse	Communication manager prepares the response on behalf of the server using this method
addResponse	It add the response in appropriate format in preparation for sending it to the server
getResponse	The *callbackhandler* uses this method to receive the response after it gets a message from the communication manager that the response of the request is ready for delivery.
dispatchResponse	This method is used to deliver the response to the client.
disconnect	Disconnects the communication between the client and the server.

On the other hand, *the CommunicationManager* can also provide synchronous communication between the client subsystem and the server subsystem, especially in situations where users make explicit request for recommendations. In this case, the *CallbackHandler* services would not be required, the client subsystem sends the request through the *CommunicationManager* to the server subsystem, and the client waits for the response.

It then waits to receive the response from the Recommendation Manager. When the recommendation list is ready, the recommendation manager sends it to the communication manager, which then sends a notification to the callback handler. It then pulls the recommendation list from the communication manager and sends it to the client subsystem. It handles recommendation request, for example, whenever there is a change in the user's contextual situation, through methods such as *setRecognizedContext()* of the *ContextRecognizer* class of the client subsystem. It then includes the user's current contextual information in the *getRecommendation()* method to the server subsystem. The client service then sends a message to the *CommunicationManager*'s *CallbackHandler*, to create a callback that is responsible for retrieving the response from the server for the request sent by the client. It then prepares the HTTP connection between the server and the client. After connection has been established, the client then sends the request to the *CommunicationManager*, which then prepares the request and dispatches it to the server subsystem. The same process is repeated from the server end when the server is ready to send the response.

5.8 CAMR integration and possible implementation

The design of CAMR as presented in this chapter is implementation platform neutral. Thus, as depicted in Figure 5.27 the emphasis of the design is not on the implementation programming language or platform but rather on the functionality provided by the proposed solution. Bearing this in mind, the design groups a set of objects into modules, with each module providing functionality or a group of related functionality, which are exposed as Web services, allowing interoperability among those modules and between CAMR and external services. Essentially, different parts of the system can be implemented based on different hardware or software platform hence communication between its various parts can be realized using web service interfaces, to allow platform independent implementation of the framework. For instance, it is expected that the mobile subsystem can be implemented using any mobile platforms such as Android, Windows Mobile, or as Web client because the services of the framework are exposed as Web services, while services such as context awareness provided by the client are also provided as web services. Figure 5.28 illustrates at package level, the modules of both the CAMR client and server subsystems.

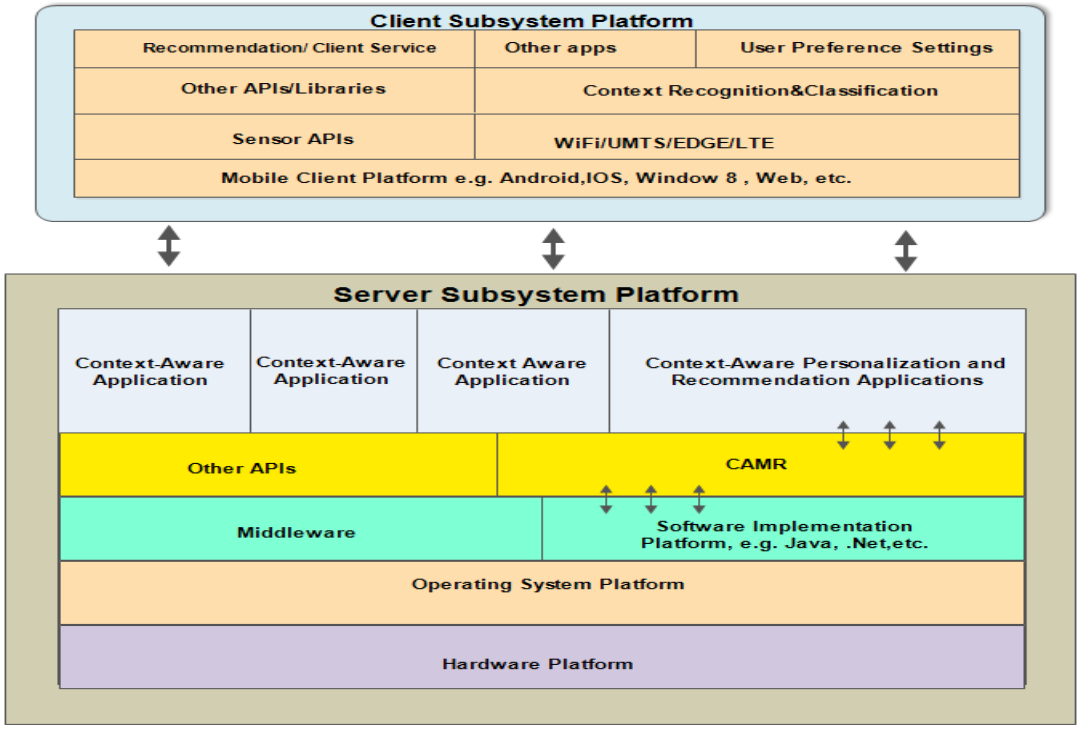


Figure 5.27 - CAMR High-level software and hardware platform-independent integration

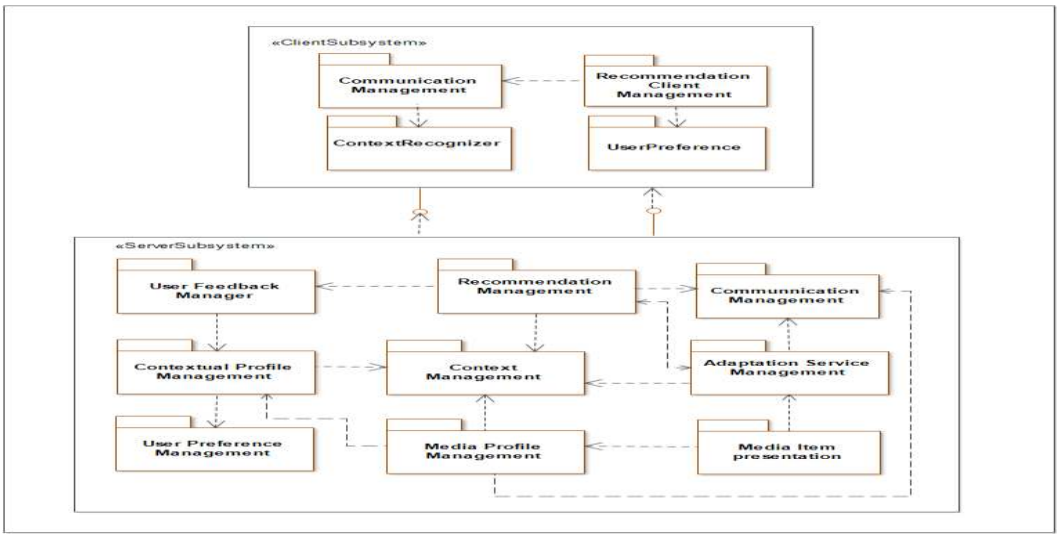


Figure 5.28 - CAMR package intra and extra communications between various modules.

For example, the client and server modules communication via web service interface by exposing important functionality such as provisioning context resources as web service resources. There are many popular and versatile platforms to implement the proposed solution. Python, .Net, Java EE to mention a few as long as such platform supports object orientation and web services.

However, for evaluation purposes, an experimental proof-of-concept of CAMR was implemented based on Java technologies. The server subsystem components were implemented in Java EE 6 (Java platform, Enterprise Edition version 6). Java EE 6 provides powerful sets of APIs that reduces application's complexity and development time. It presents an application model that defines the architecture's services as multitier applications for delivering scalability, accessibility, interoperability, and manageability [185]. It also provides standard system level services, thereby resolving complexity usually encountered when developing multitier or multilayer applications.

Java EE is naturally a distributed platform, with application logic divided into components according to the functionality provided by such components, and these components could be installed in different machines depending on the tier in the multitier environment to which such components belong. Between each component and underlying system or low-level services, are the containers, which provide interfaces between each component and the low-level services and platform-specific functionality that supports such components. Because of the platform independent nature of Java EE, it was adopted as the platform to implement the design presented in this chapter. On the other hand, the client was implemented as an Android application. There are other excellent alternative mobile platforms such as iOS, Windows, etc. However, we chose Android platform because it provides a complete sets of mobile application framework that allow rapid development of innovative applications on the latest smart devices such as smartphone, wrist watches, TV, and even cars[186]. Android has the largest share of the mobile markets and it is forecast that the sales of Android run devices will hit nearly one billion (950.5 millions) in 2014. Similarly, Android has support for Web services implementation, coupled with its flexible and easily programmable APIs, with access to device's hardware and sensors in particular, for this reasons and others, we implemented the client subsystem of the framework for experimental purpose on Android smartphones.

5.9 Summary

This chapter of the thesis elaborates on the analysis and the design of a software system supporting context-aware media recommendations in a mobile environment. The design and the analysis follow software engineering principles, first by analyzing a usage scenario from which functional and non-functional requirements of the system are elicited. In the second phase, the functional requirements derived from the scenario were then used to develop a use case model, which abstracts, at a higher level, the functionality of the proposed system, including the algorithms and concepts discussed in chapters 3 and 4 of the thesis. A conceptual, functional architecture of the system, including, its classes, packages, as well as models representing their interactions and important operations supporting the provisioning of personalized multimedia services have been elaborated and presented. In the next chapter, evaluation of the proposed CAMR framework is presented via a context-aware recommendation application, a concrete experimental implementation, which was developed, based on the concept of the design presented in this chapter, and the processes presented in the preceding chapters. The application serves as a platform to evaluate the efficacy of the framework as well as for conducting controlled experiments including a user evaluation study.

Chapter 6

Experimental Evaluation

6.1 Introduction

This chapter describes the evaluation work that was carried out during the course of this thesis. It presents two sets of experiments that were conceived, the procedures taken, and the results obtained. An empirical approach was adopted, whereby test data sets were obtained from real usage conditions to experimentally evaluate the system and thus assess the validity of the proposed conceptual approach as well as the feasibility of its implementation.

The first set of experiments, presented in the first sections of the chapter, deals with the evaluation of the context recognition service, as part of the context-aware personalized recommendation framework. The ability of the framework to recognize a mobile user's dynamic contexts correctly, using information gathered by the handheld device was evaluated. This evaluation goes from assessing the most suitable sensors available on the mobile device, through the type of features to extract from data acquired by those sensors, to the performance of different classification models for processing such extracted data. In particular, it analyses the performance of different models when used to process time-series features extracted from data obtained from accelerometers, rotation, and orientation sensors, providing an indication of the most appropriate ones to recognize contexts.

The second set of experiments deals with the overall evaluation of the proposed context-aware personalized recommendation framework. These experiments were conducted to evaluate the efficacy and feasibility of the contextual recommendation services via a proof-of-concept context-aware personalized recommendation application. In this set of evaluations, two categories of experiments were performed: data driven and user centered experiments. They both work with data acquired in real usage conditions, in terms of users' media consumptions and users' contexts. In the data driven evaluations, there is no direct participation of users in terms of feedback collection. The advantage of this kind of experiment is that once the data have been collected, we can reuse them to evaluate various properties and functions of the proposed framework. On the other hand, the user-centered experiments involve real users where they are asked to express their level of satisfaction of the system based on the recommendations provided to them. Essentially, this latter category of experiments involves collecting feedback from users and using the feedback to obtain a measure of the efficiency of the system.

Before describing the validation and evaluation procedures, this chapter starts by discussing the challenges of evaluating context-aware personalized recommendation systems. It then presents some assumptions upon which these evaluations were designed as well as the scope of the evaluations.

6.2 Challenges of developing and evaluating context-aware media recommendations

Ultimately, real users under normal and real-world conditions should evaluate systems meant for human use. However, as we know, this is impractical most of the time and thus simulations are generally used instead. Sometimes simulations can even outperform the efficacy of real experiments as they allow testing the system under different conditions, which might occur seldom in real-life. Eventually, the ideal solution is to use both simulation and real experimentation. The evaluation of context-aware recommender systems imposes additional challenges given that their response is expected to adapt according to the consumption conditions (the context!) and the feedback of the user (either explicit or implicit). Accordingly, it becomes much more difficult to model the expected behavior of the system. We have thus decided to implement a prototypical instantiation of the proposed conceptual framework together as a case-study application, notably, a context-aware personalized recommendation application, for suggesting movies. As

discussed in chapter 5, the implemented system adopted client-server architecture, with its client subsystem specifically developed for the Android OS and deployed on an Android smartphone.

The user terminal could nevertheless be other kind of multimedia-enabled mobile device, namely laptops, smartphones or tablets, as long as it is able to offer sensing capabilities. Both the client and the server make their functionality available as services either to each other, or even to external third-party applications. The client gathers contextual data, triggering the recommendation process, whose main operations run on the server side. It displays the recommendations received from the server and presents selected media contents to the user. The server subsystem was implemented on a separate machine with higher capabilities to support the computational requirements of the framework's functions. Before actually describing the experimental procedures used, this chapter discusses the problems associated with such procedures and explains the scope of the conducted evaluation.

6.2.1 Aspects to consider for evaluating context-aware media recommendation systems

One of the key requirements imposed on context-aware recommendation systems is that such systems are able to dynamically acquire the user's contexts, so that they can automatically adapt their behavior or response accordingly, modifying the set of recommended items as the context changes. The context awareness model's primary function is the timely acquisition of user's contexts and communication of such contexts to the recommendation engine to trigger the recommendation process. However, evaluation of such systems is not a trivial task, especially in what concerns the gathering of test data, considering that mobile users' contexts change rather frequently. Thus, obtaining user's preferences in every possible context becomes almost impossible. Therefore, it is important to conduct experiments in a controlled setting, scaling down the number of contextual situations used for evaluation purposes.

Work reported in literature addressing the experimental evaluation of context-aware systems, usually relies on basic contextual information about the user, namely personal or demographic data such as, sex, age, nationality, etc. [1], [135], [187]. Such static contextual information can be considered as quite limited to test the ability of the system to react dynamically to context changes and would thus lead to a rather poor evaluation. Our experimental procedure effectively considers the acquisition of more dynamic contextual data such as user location, weather information, user activity, etc. Please note that we use context and activity sometimes interchageably to mean the same thing. User's activities such as walking, sitting, jogging, etc. are subset of contextual information.

Another known problem is the issue of privacy and trustworthiness. Many users are quite willing to disclose private information and hence may not be willing to allow third parties access to e.g. their location, their content consumptions or the activities they perform. In fact, privacy and related issues are some of the difficulties that are commonly encountered when evaluating context-aware applications.

Finally, there is the problem associated with time. Evaluating context-aware systems in real-life scenarios is an expensive and time-consuming exercise [1]. It is therefore difficult to persuade people without adequate compensation to give up their daily activities in order to evaluate the system in every possible contextual situation. To overcome this limitation, some evaluations use only simulated contextual data. However, this solution has the significant drawback of making it impossible to collect the feedback and opinions of real users. Not to mention the large number of parameters and variables that would need to be modelled. Additionally, the contextual situations that are simulated are those that have been foreseen by the researcher and will thus be biased by his/her experience. They will most certainly lack the diversity of natural user's behaviors in the face of usual or unusual situations

A possible solution is to develop an application that can collect contextual information of voluntary users in a natural setting and then using the collected context data offline for evaluation purposes. This method would allow collection of user's contextual information in a way that is close to the natural application scenarios of the system.

Therefore, considering the stated problem above, a case study application was implemented to obtain user's information in both offline and online modes. In an offline mode, the system collects data and pre-processes the data

before using it to provide recommendations. In an online mode, a more expensive mode, the system collects data during evaluation for the recommendation process.

The case study application is a context-aware personalized movie recommendation service: it offers the user a GUI to select and consume video and, using the prototype implemented according to the devised conceptual framework. It automatically performs all of the following operations: it registers the user's consumptions by analyzing and classifying the consumed content; it has the capability to gather contextual data, recognizing the activity that such user performs, using part of those contextual data; it applies reasoning mechanisms to infer high-level context; it builds the contextualized user profiles; it selects from candidate movies, the ones that suit better the current context of the user, delivering to the user a set of recommended movies. Thus, to evaluate the proposed framework, the contextual information, including respective user preferences were acquired offline in order to be able to evaluate the critical properties of the proposed system.

6.2.2 Evaluation scope

As stated above, evaluating context-aware recommendations is a complex and time-consuming process. Therefore, executing exhaustive evaluation of all possible contextual scenarios is not practically feasible considering time and resource constraints as well as other user factors. In this regard, the evaluations of the proposed framework were executed in a controlled experimental setup by asking a group of users to collect contextual information using the CAMR client application and associating their multimedia consumption preferences to the corresponding context information.

In the experiments, the context collection process serves two purposes. First, acquired contextual data were used to allow evaluating various classification models whose objective is to recognize first, the activity that the user performs and second, the activity and location where such activity is performed. This first step had as its primary goal, which is the selection of the best classification mechanism to be incorporated into the CAMR context framework. Different activity context recognition models were then evaluated using the acquired context data to ascertain their effectiveness and accuracy and the best one was then selected and used. These experiments did not allow evaluating the performance of the recommendation system as a whole, under normal operating conditions, but rather for selecting the best activity recognition approach to use. Accordingly, the results obtained with the first set of experiments reflect the performance of the system concerning its ability to recognize user's activity contexts using contextual data acquired by some of the smartphone' s built-in sensors.

The goal of the second set of experiments was to assess the performance of the CAMR framework to deliver recommendations adapted to the recognized usage context. Although the system was built with the necessary mechanisms to infer additional characteristics of the usage contexts other than the activity the user performs (using ontologies and reasoning), the conducted evaluation did not address those aspects. Many researchers have presented works that evaluate such context-awareness properties [26]. As such, we have concentrated on inspecting the benefits that could be achieved by the recommendation system, from the user perspective, when knowing the type of activity the user was conducting.

Summarizing, the objective of the evaluations that were conducted and that are described in this chapter, were principally to validate the adequacy of using some of the sensors built-in the mobile device to recognize the user activity contextual situations accurately, and to evaluate the capability of contextual information to improve the quality of personalized multimedia recommendations. Essentially, the aim is to evaluate if the use of contextual data effectively brings benefits when recommending items to mobile users. Obviously, to arrive at meaningful conclusions, it was also necessary to provide indications concerning the quality of the context recognition sub-system.

6.3 Context recognition evaluation

In chapter three, processes and concepts required to build automatic context recognition mechanisms of the context awareness framework on mobile devices were presented. This section describes the actual complete context classification process, implemented for evaluation purposes.

Accordingly, instead of adopting the common approach of explicitly acquiring contextual data to obtain a training set, we decided to collect contextual data with associated user preferences. This way, in one-step, it would be possible to collect the data for two types of evaluations: the context recognition and the context-aware recommendation evaluations. Specifically considering the former, we have developed a sub-system suitable for mobile devices as part of the CAMR client whose design was described in chapter 5, to build an automatic context acquisition, processing, classification and recognition model.

To realize this objective, three important steps were executed as follows. The first one involved activating the required in-built sensors on the device to acquire data from them; the second one involved the actual acquisition of data from those activated sensors. Note that we have implemented context modules as parts of the client subsystem of CAMR, including the context sensors, monitors, aggregators, classifiers, and recognizer modules, which are responsible for capturing raw sensor data from the Android handheld and processing the data to provide contextual data as high-level contextual information as explained in chapters three and five. However, we needed to evaluate various classification models using the collected context data, as well as determining the best features as explained in chapter three, that could provide better context recognition quality. Thus, the second process involved building and evaluating context recognition models using the WEKA framework and the collected data in an offline process.

6.3.1 Context recognition evaluation objectives

The main goal for evaluating the context recognition of the proposed framework is to examine the capability of mass-marketed handheld devices to provide just-in-time contextual information for contextual recommendations. To realize this goal, specific objectives were established for the experiments, designed to validate important hypotheses on the suitability of handheld device's in-built sensors to provide high-level contextual information:

6.3.1.1 Suitability of handheld device's built-in sensors for recognizing mobile user's common activity contexts. We wanted to evaluate the suitability of mass marketed mobile devices for recognizing simple activity a user performs. In addition, the thesis evalautes the following:

(a) Suitability of handheld devices' built-in sensors other than accelerometers for activity context recognition. We wanted to examine the suitability of sensors other than accelerometers for context recognition. In literature, most authors realized context recognition using only accelerometer, which uses device's movement to determine user's context[100]-[101], [103]. However, in practice, data from other sensors could contribute more meaningfully to accurately determine user's contextual information.

(b) Evaluation of combination of smartphone acceleration, rotation, and orientation to determine if using more than one sensor can improve recognition performance. This evaluation was designed to understand the benefits of combining data from more than one sensor to obtain high-level contexts, considering the fact that activating many sensors on the device could qickly lead to battery drain.

6.3.1.2 Evaluation of various classification models that could provide accurate context recognition performance using mobile device's in-built sensors. We wanted to compare performances of a number of classification models for context recognition tasks. This would help to make informed decisions on the classification models that are best suitable for on-device context recognition in terms of recognition accuracies using our data. The results of these evaluations could be useful to developers for taking informed decisions when building contextual recommendation applications.

6.3.1.3 Evaluation of a combination of time series features used by classification models as inputs for context recognition. The objective here was to determine the best time series features that can be used to extract more accurate high-level context information from low-level context data obtained from device sensors. It was decided to evaluate only time series features because other types of features such as those based on Fast Fourier Transforms (FFT) have been confirmed in the literature to be computationally expensive and therefore are out of the scope of the proposed experiments [116], [188].

6.3.2 Context recognition experimental setup

To realize the objectives of the context recognition evaluation enumerated in the last section, in this section, the thesis presents the context recognition experimental setup, the choice of software and hardware platforms, as well as context recognition processes, starting from the context data collection, preprocessing, feature extractions, and recognition of high-level context using classification models. Please note that details of these processes and models have been presented in chapter 3 and will not be repeated here.

6.3.2.1 Choice of mobile platform and sensors

To evaluate context-aware mobile systems, the choice of the device platform on which to demonstrate their applicability is very crucial. In this thesis, we define a device as any computing platform used by mobile users i.e. a device that could move easily with the users as they move from one place to another. These devices include laptops, tablets, mobile phones, or even iPods, and wristwatches. However, the evaluation will focus only on mobile phones and tablets to demonstrate the feasibility of the proposed framework. The reason is that mobile users are always with these devices and thus, it is natural to notify them of any recommendations on this type of devices. It makes little or no sense, for example, to send recommendations to the user's laptop when he or she is jogging in the sport complex or when she is driving to the office. Another reason is that users do not always move around with some devices such as laptops, and therefore, contextual information such as user activity would be difficult to capture. On the other hand, mobile phones and tablets come in diverse operating platforms. There are billions of mobile devices running on diverse hardware and software platforms. These mass-marketed devices operate on popular mobile OSs, such as iOS, Windows mobile, Symbian, Android, etc. In terms of implementation, one of the features that set Android mobile platform apart from others is the ease of prototyping innovative ideas and the availability of rich, well designed, and powerful APIs to execute diverse functionality, with greater flexibility. Furthermore, availability of diverse and easily programmable embedded sensors, with well-designed open source APIs is another important feature that makes Android platform a good choice for experimentation with context-awareness. These APIs are designed to access easily low-level hardware, such as device's in-built sensors.

Thus, this open platform nature of Android makes it the mobile platform of choice to experiment with innovative ideas because it can be customized easily to effectively minimize the need for providing additional external hardware sensory resources when compared to other platforms such as Symbian and iOS. Another attractive feature of Android devices is the fluid and attractive user interfaces it offers, with ease of design and implementation, though still limited by size compared to desktops or laptops. Similarly, Android provides services that can run in the background; this allows monitoring of contextual information while users are busy using their devices for other purposes, thereby helping to minimize user's frequent interventions in the course of automatic gathering and processing of contextual information for recommendations. Thus, for these reasons we considered only devices running the Android operating system [112], though the design of the CAMR, as presented in chapter 5, can be implemented on other mobile platforms such as iOS, Microsoft Windows 8, etc. In this regard, CAMR context recognition sub-system, or application, was deployed on a Samsung Galaxy S I9000 smartphone running Android 3.2 (Honeycomb) operating system with the objective of automatically acquiring natural contextual data and subsequently use those data to recognize the contexts of the user.

6.3.2.2 The taxonomy of contexts to be recognized

To collect handheld device sensor data that represent mobile user's basic activity contexts, we defined two different taxonomies to classify this kind of contexts, which are distinguished essentially by the use or not of location information. The first taxonomy uses location to identify the activity and for that reason it classifies them using both physical activities as well as the locations of the user(e.g. "*sitting/standing at home*", "*sitting/standing in office*", "*sitting/standing in a bus*", "*driving a car*", "*sitting/standing @ trainStation, etc.)*. The second taxonomy does not use location information but classifies contexts simply based on physical activities , examples are: "*walking*", "*jogging*", "*lying*", "*running*", "*standing*", "*driving*", and "*sitting*". In this form of classification, more detailed/complex description are also foreseen, namely "*ascending stairs*", "*descending stairs*", "*ascending in elevator*", and "*descending in elevator"*. These activity contexts and locations were selected because they represent typical daily contexts of an average mobile user. These contexts induce different accelerations, rotations and orientations along their three axes (X-axis, Y-axis, and Z-axis).

6.3.2.3 The context data acquisition process

As earlier stated, the application was developed on Android mobile platform [189]. Figures 6.1, 6.2, 6.3 and 6.4 show the application visualizations. The application implements a background service, which runs on a single background thread, allowing executing its operations without being interrupted by other operations of the device when data collection process is in progress. Figure.6.4 shows a sample of captured low-level sensor data. The application, through these simple interfaces, provides the capability that allows users, during data acquisition process, to select from a list of activities, locations, and other context information on the smartphone screen. While users perform these activities carrying their devices, the application gathers and logs all sensor data corresponding to the selected context information. They could start the application, allowing it to run in the background to track longer activities.

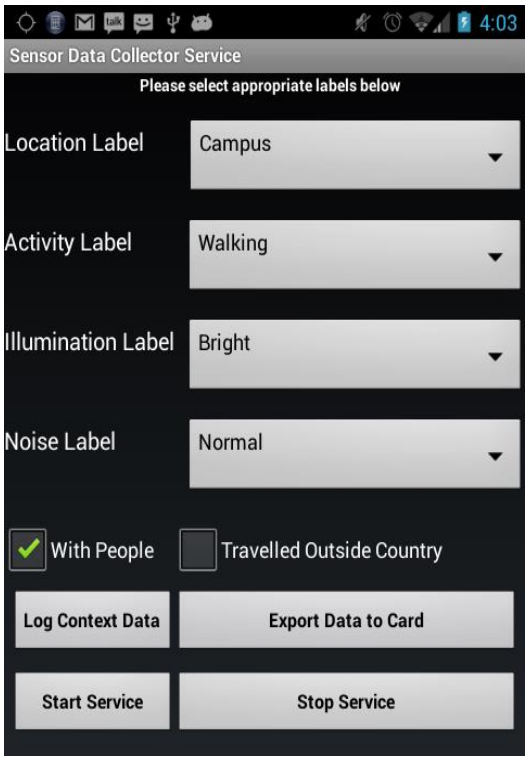


Figure 6.1- User context data collector interfaces

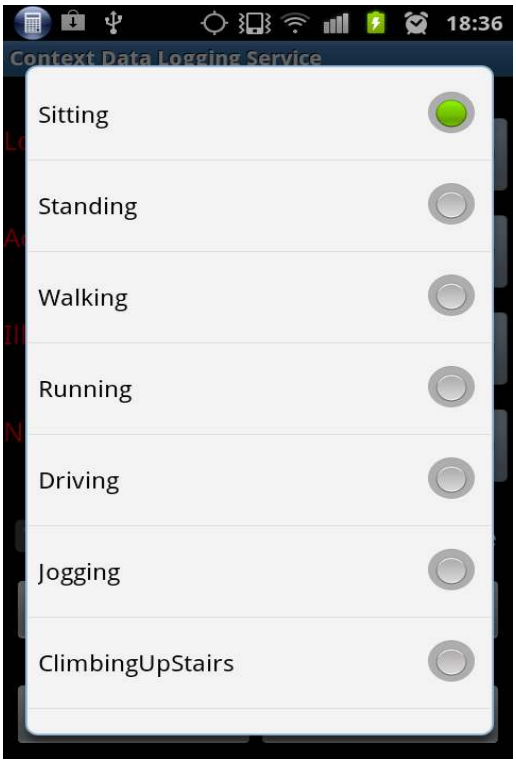


Figure 6.2- Logged low-level smartphone sensor data

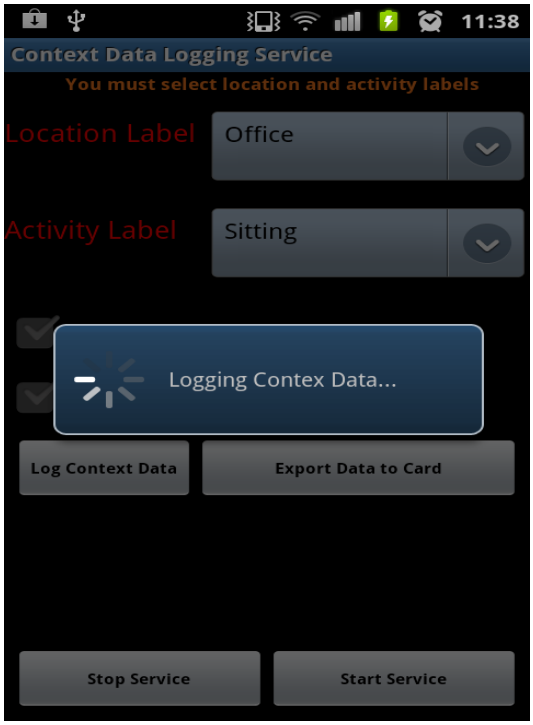
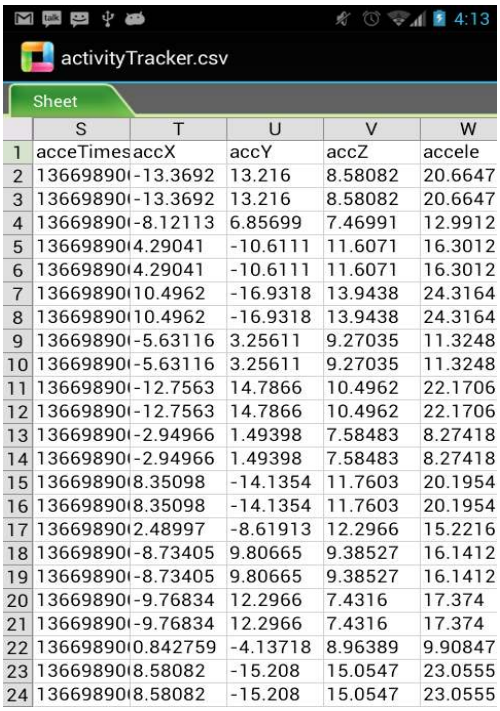


Figure 6.3 -Logging low-level context



	S	T	U	V	W
1	acceTimes	accX	accY	accZ	accele
2	13669890(	-13.3692	13.216	8.58082	20.6647
3	13669890(	-13.3692	13.216	8.58082	20.6647
4	13669890(	-8.12113	6.85699	7.46991	12.9912
5	13669890(	4.29041	-10.6111	11.6071	16.3012
6	13669890(	4.29041	-10.6111	11.6071	16.3012
7	13669890(	10.4962	-16.9318	13.9438	24.3164
8	13669890(	10.4962	-16.9318	13.9438	24.3164
9	13669890(	-5.63116	3.25611	9.27035	11.3248
10	13669890(	-5.63116	3.25611	9.27035	11.3248
11	13669890(	-12.7563	14.7866	10.4962	22.1706
12	13669890(	-12.7563	14.7866	10.4962	22.1706
13	13669890(	-2.94966	1.49398	7.58483	8.27418
14	13669890(	-2.94966	1.49398	7.58483	8.27418
15	13669890(	8.35098	-14.1354	11.7603	20.1954
16	13669890(	8.35098	-14.1354	11.7603	20.1954
17	13669890(	2.48997	-8.61913	12.2966	15.2216
18	13669890(	-8.73405	9.80665	9.38527	16.1412
19	13669890(	-8.73405	9.80665	9.38527	16.1412
20	13669890(	-9.76834	12.2966	7.4316	17.374
21	13669890(	-9.76834	12.2966	7.4316	17.374
22	13669890(	0.842759	-4.13718	8.96389	9.90847
23	13669890(	8.58082	-15.208	15.0547	23.0555
24	13669890(	8.58082	-15.208	15.0547	23.0555

Figure 6.4 - Logged low-level smartphone sensor data

They could also stop the application when not in use to conserve device's battery. Thus, we collected data using the client side application in the following process. Six individuals volunteered, and participated in the data collection exercise carrying the handhelds while performing the defined activity contexts. The volunteers selected the labels corresponding to the activity contexts they were about to perform. The application then logged (see Figure 6.3) the sensor data while the selected activity was being performed. The data from the rotation vector, orientation, and accelerometer sensors, in three axes, were all gathered and logged including data from the GPS, proximity sensor, magnetometer, and light sensors. The application was designed to take 10-second long chunks of data, resulting in various (64,128, 256,512, and 1024) data points contained in each axis of the sensors, overlapping at 50% between consecutive data points.

Data from other sensors such as GPS come in two dimensions (longitude and latitude); whereas sensors like light, microphone, and temperature sensors have one stream of data. The accelerometer X- axis's positive and negative readings represent the right and the left horizontal movements respectively. Its Z-axis's positive and negative readings represent the forward and backward movements respectively, whereas its Y-axis's positive and negative readings represent upward and the downward accelerations respectively. Recording the low-level sensor data requires the determination of the sensor's sampling rates. However, the Android platform does not support sampling rate. Instead, Android uses event triggering whenever values of the sensor events change [112]. The rate of events in Android can be set to one of four thresholds - UI, Fastest, Normal, and Game [112]. Fastest is ~ 40Hz, whereas normal is the slowest with ~4Hz.

We set the event-triggering rate at fastest to capture every event during sensor recording. In addition to the accelerometer, the rotation vector and orientation sensors, latitude and longitude data of the GPS were also logged. The collected data were exported as CSV (Comma Separated Values) file from the phone SQLite database to the Secured Digital (SD) card for further processing. The preprocessing phase via context aggregator converts all the

sensor data obtained from the monitoring phase into samples to identify basic contextual features. After collecting various labelled contextual information, to evaluate the models, the data were then processed using the WEKA framework to build the recognition model. Statistical features were generated from the collected context data, which were then fed into classification models for training. On one hand, we used cross validation approach for testing. On the other hand, we used separate datasets for training and testing in the evaluations. The primary goal of this process is to determine the generalized performance of each algorithm to ascertain the ones with the best performance for building the context prediction model. To build the context recognition models, we needed to evaluate a number of classification models, with the aim of identifying the models that recognize each context being investigated with good quality. The models include well-known base level classifiers that have been used in existing context and activity recognition projects [170]. The following are the algorithms that we evaluated: kNN, K-Star, Decision Trees, Naïve Bayesian Network, SVM, MLP Logistic Regression, RIPPER: Repeated Incremental Pruning to Produce Error Reduction and PART: Projective Adaptive Resonance Theory.

Table 6.1 - Summary of popular algorithms used in context recognition

Algorithms	***Reference***
KNN	[111],[118][169]
KStar	[190][104]
LibSVM,	[118]
C4.5	[97],[103],[107],[114][168]-[169]
Random Forest	[101],[168]
SVM	[118],[214],[169][204]
MLP	[97][102] [104]
Logistic Regression	[97][115]
RIPPER	[169]
PART	[103]
Naïve Bayes	[104] [118] [114] [169]

Table 6.1 presents the summary of these models and references to existing projects that used them for activity recognition. Readers who are interested in their full descriptions can check [166], [168], [170], [171].

6.3.2.4 Context recognition evaluation metrics

The conducted experiments allowed collecting smartphone in-built sensor data using test users in different location in Porto performing various selected activity contexts. Such collected data were labeled and processed to extract between 30 and 90 statistical features adoption the methodology explained in chapter 3 section 3.4. First, we divided the data into two main sets. The activity context data set without location and activity contexts with location dataset. For example, activities such as "*sitting in a bus/train*" and "*sitting at home/in office*" are not the same because the patterns of events generated by the sensors during these activities are significantly different. Therefore, it is important to know where a user performs an activity, such as sitting or standing. We have used the location information to disambiguate such activity contexts. In addition, we divided each of the two datasets into five sub-datasets, with window lengths of 64, 128, 256, 512 and 1024 samples (approximately 1.5, 3, 6, 12 and 24 seconds respectively). These experiments are important because we needed to analyze the models based on these parameters. This process guided the choice of model selection, the choice of suitable features and the choice of window length that are suitable for implementation on the mobile device. The Weka data mining tool was used for model building, training, testing and evaluation [172]. For testing purposes, we have used two approaches described in (a) and (b) as follows.

(a) *N*-fold cross validation: The cross validation technique uses a pool of data drawn and split into *n* sets of approximately equal size and class distribution. For each fold, the classifiers are trained using all but one of the *n* groups and then tested on the unseen group [104].

This procedure is repeated for each of the *n* groups. It is a commonly used technique employed to guide against models' over-fitting and it is good at estimating generalized performance of different classifiers [176]. Many previous works adopted this technique [97], [99]-[105]. The main idea is to train the models on a subset of the data and then evaluate them on the part of the data not seen during training.

(b) User independent evaluation: In this approach, test set consists of data of individuals who are not included in the model training process. In other words, it involves training on one set of user data and testing on another set of data of unrelated users. This is to ensure user independence of the models. Thus, for this type of evaluation, data from all users were used for training except a user, whose data were used as test dataset and this was then repeated for each user data. This process is called leave-one-subject-out validation [107].

(c) **Evaluation metrics:** The metrics used in the context recognition experiments are, precision, recall, F-Score, Root Mean Square Error (RMSE), and confusion matrix. These are the most widely used metrics to evaluate context recognition models in the literature [190]-[191].
F-Score is an often-used metric in information retrieval and natural language processing communities and it is interpreted as the weighted average of precision (*P*) and recall (*R*). It is a measure of the statistical accuracy of the models given as follows:

$$\textbf{\textit{F-Score}}(R, P) = 2*RP/(R+P) \quad (6.1)$$

Where recall (*R*) is the measure of the ability of a classification model to select instances of a certain class from a dataset. It is the sensitivity of the model defined as:

$$R = TP/(TP+FN) \quad (6.2)$$

TP is the number of true positive predictions and *FN* is the number of false negative predictions. Precision *(P)* is the measure of the accuracy if specific class is predicted; defined as:

$$P = TP/(TP+FP) \quad (6.3)$$

FP is the number of false positive predictions.

Root Mean Square Error (RMSE) is the measure of the difference between values predicted (p_i) by the models and the actual values (q_i) observed. Smaller values of RMSE indicate higher and better predictive model's performance.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - q_i)^2} \quad (6.4)$$

Note: The WEKA data mining tool used in the offline context recognition experiments uses quadratic loss function to calculate RMSE to evaluate classification models using the probability that a given test instance belongs or not to a particular class. For details, interested readers can check page 158 of [172].

Confusion matrix is a square matrix of order *n* number of classes used to present detailed results of a multiclass classification problem [168].

The confusion matrix gives a more detailed and fine-grained analysis of both correctly and incorrectly classified classes of the supervised learning based models. A given element $c_{i,j}$ of the matrix is the number of instances belonging to class *i*; classified as class *j*. Information about classification errors is also presented by the confusion matrix.

6.4 Context recognition experiments

In this section, the experimental evaluations of the context recognition models are presented.

6.4.1 Suitability of mobile device sensor for context recognition

To examine the suitability of handheld devices such as smartphone sensor data for context recognition task, we have taken into consideration the importance of window lengths [114], [169], [192]. While we examined the suitability of handheld device's sensor data for context recognition, we also examined the effect of window lengths on context recognition accuracies of the models. We wanted to determine the window lengths that produce good performance. In the experiment, we varied the window lengths using 64, 128, 256, 512, and 1024. (Table 6.2 shows details of the window lengths, time in seconds and number of data samples in each window).

Table 6.2 - Window lengths and number of class instances

Window Size	*Time(Seconds)*	*No of Instances*
64	1.5	7084
128	3	3537
256	6	1763
512	12	876
1024	24	433

As shown in Figure 6.5, at least five of the models produced low recognition error based on RMSE with window length of 128, while the recognition error varied for 64, 256 or even 1024 depending on the model. kNN has the lowest RMSE and thus, highest recognition rate with each window length, closely followed by Random Forest. The result shows that classification models can accurately recognize user contexts using smartphone built-in sensors data. On the other hand, judging by the effect of sliding window length on the recognition accuracies, our experiment shows that window length of 128 is the best for our data. We observe that some models performed better with larger window lengths, whereas others performed worse and vice versa. For example, while C4.5 produced low to high RMSE with smaller and larger window lengths respectively, we observed that for window lengths of 512 and 256 samples, the RMSE are 0.122 and 0.09 respectively. Nevertheless, it is evident from the result that nearly half of the model produced good recognition performance with shorter window lengths of 64 and 128 samples. This result agrees with the conclusion of Stefan et al. in [104].

Additionally, in Table 6.2 we observe that the number of instances is inversely proportional to the window lengths. This has significant impact on the recognition performance of the models as revealed in Figure 6.6, using kNN model. The larger the window length, the poorer the recognition performance. This relationship between window length and number of instances affects the distribution of the data for selected contexts and thus their recognition by the models.

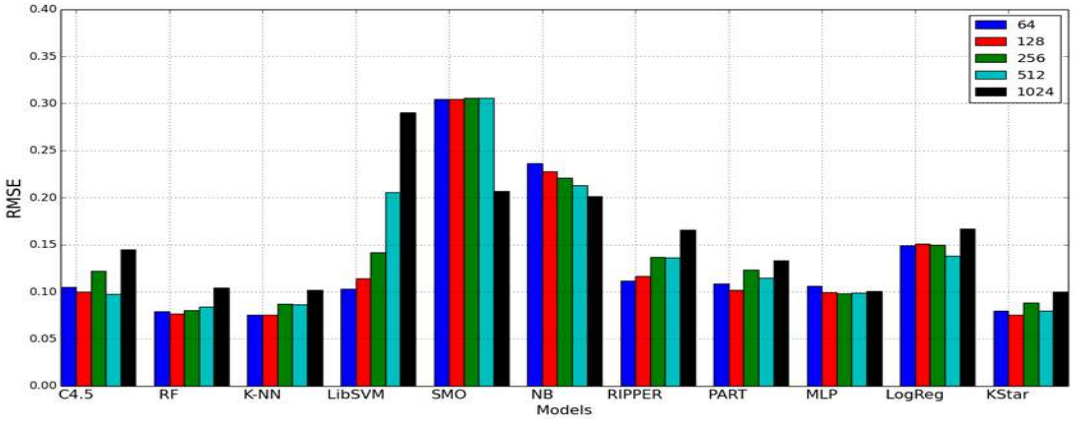


Figure 6.5 - Classifiers' recognition RMSE for selected window lengths

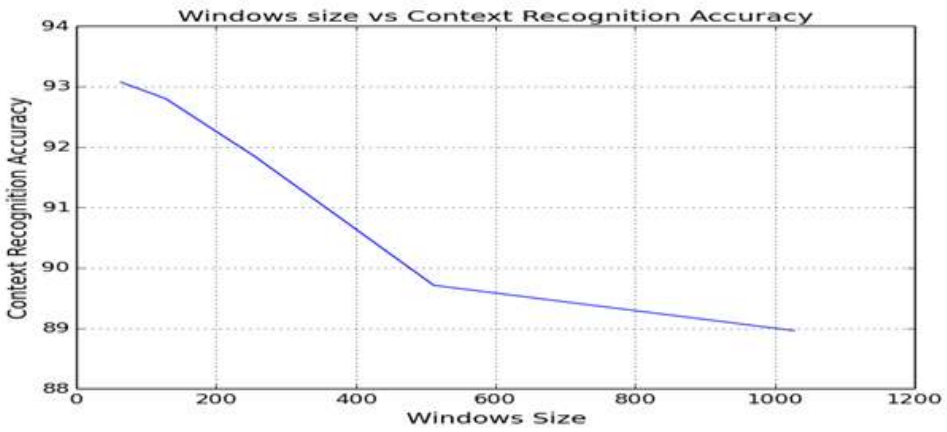


Figure 6.6 - Window length versus context recognition performance

6.4.2 Feature selection evaluation

In this experiment, the capability of time series features for recognizing activity contexts was analyzed. The goal of this experiment is to determine the recognition accuracy of each context, using feature/algorithm combinations. This was done to identify features that are suitable for recognizing context information. Additionally, the aim of this experiment is to identify redundant features on one hand and on the other hand to identify those features that provide additional recognition accuracy improvement for each model. Each feature was analyzed using the data with 128-window length with 50% overlapping. Table 6.3 presents details of features and models as well as the contexts recognized accurately. Numbers **1** to **7** in the table represent ***lying, driving, jogging, running walk, sitting***, and ***standing*** respectively whereas ***8*** represents all the contexts. The result shows that decision trees (C4.5 and Random Forest), kNN, RIPPER and PART could recognize all the contexts, using either maximum, minimum or range features. This result shows that these contexts can be recognized with simple and minimal number of feature set, which require little computational cost. The result shows simple features such as that range can recognize the activity contexts. This result confirms the conclusion of similar evaluations of these simple feature sets that they are good enough to recognize activity contexts with minimal computational cost [107].

Table 6.3 - Activity recognition capability of each time domain feature and classification models

Numbers 1 to 7 in the table represent *Lying, Driving, Jogging, Running Walking, Sitting,* and S*tanding* respectively whereas 8 represents all the contexts.											
Features	***Algorithms/Activities Correctly Recognized***										
	C4.5	**RF**	**KNN**	**KStar**	**SMO**	**LibSVM**	**NB**	**MLP**	**RIPPER**	**PART**	**Logistic**
Max	8	8	8	1,2,3,4,5,6	1,3,4,5,6	1,2,3,4,5,6	1,3,4,5,6	1,2,3,4,5,6	8	8	1,3,4,5,6
Min	8	8	8	8	3,4,5,6	8	3,4,5,6	1,2,3,4,5,6	8	8	3,4,5,6
Med	1,3,5,6,7	1,2,3,5,6,7	1,2,3,5,6,7	1,3,5,6,7	5,6	1,3,4,5,6,7	1,5,6,7	1,5,6	1,5,6,7	1,2,3,5,6, 7	5,6
Range	8	8	8	8	3,4,5,6	8	3,4,5,6	1,3,4,5,6,7	8	8	1,3,4,5,6
STD	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	3,4,5,6	2,3,4,5,6	3,4,5,6	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	3,4,5,6
Mean	1,5,6,7	1,5,6,7	1,5,6,7	1,3,5,6,7	1,5,6	1,5,6	1,5,6	1,5,6	1,5,6,7	1,5,6,7	5,6
SOS	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6,7	1,4,5,6	1,5,6	1,3,4,5,6	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6, 7	3,4,5,6
RMS	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6,7	3,4,5,6	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6,7	1,3,4,5,6, 7	3,4,5,6,7
Var	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	2,3,4,5,6	3,4,5,6	2,3,4,5,6	3,4,5,6	3,4,5,6	2,3,4,5,6	2,3,4,5,6	3,4,5,6
ZC	4,5,6	3,4,5,6	3,4,5,6	4,5	4,5	4,5,6	4,5,6	4,5,6	4,5,6	4,5,6	4,5,6

6.4.3 Recognizing basic activity contexts

In this experiment, the performance of the models for recognizing seven basic user activity contexts is evaluated and analyzed, by computing the F-Score, RMSE and Confusion Matrices of all evaluated models. These contexts are *Running, Lying, Jogging, Walking, Driving, Sitting, and Standing*. In this experiment, location data characterizing these contexts were not included, unlike in experiment 6.4.1, because we wanted to observe the performance without including complex data such as location. F-Score for each model and the selected contexts were computed, the result is shown in Figure 6.7. For these contexts, all the models except LibSVM could accurately classify the contexts. LibSVM could not identify *Jogging* and *Running* activities. Compared with other models, it has poor overall F-Score of 0.56. In order to understand the number of instance correctly and incorrectly recognized, Table 6.4 shows, for example, the confusion matrix for kNN recognition model.

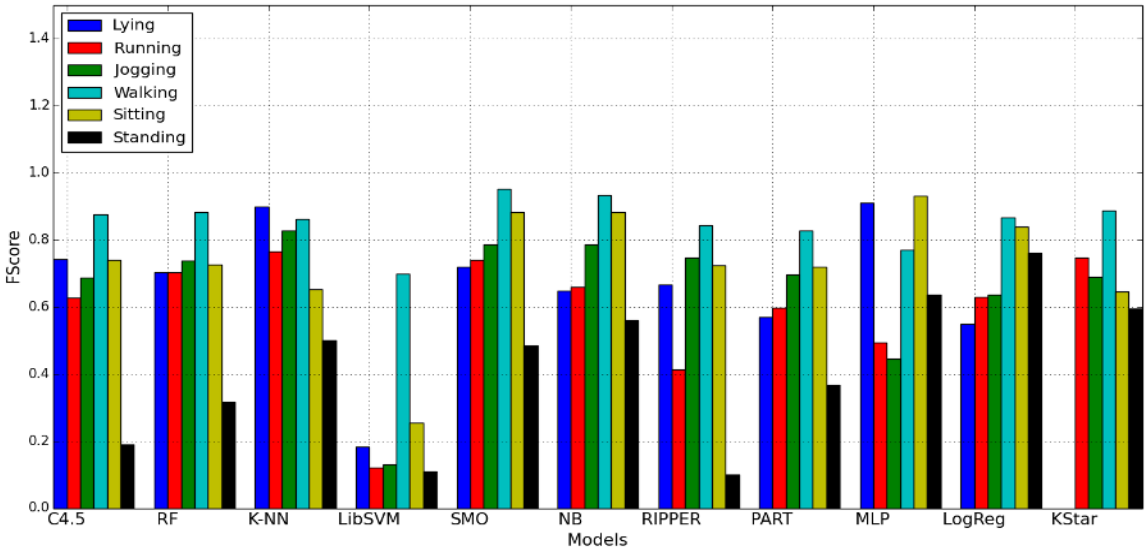


Figure 6.7 - Classifiers' FScore for selected activities

Table 6.4 - SMO confusion matrix for simple activity contexts

Predicted Class								Actual Class
Lying	Driving	Running	Jogging	Walking	Sitting	Standing		
1048	1	1	0	3	337	7	Lying	
5	**274**	0	0	17	13	8	Driving	
0	0	**239**	14	3	0	0	Running	
0	0	6	**452**	13	0	0	Jogging	
10	10	3	8	**2180**	2	0	Walking	
65	7	0	2	14	**2039**	39	Sitting	
4	8	2	0	0	1	**249**	Standing	

Table 6.5 - KNN confusion matrix for simple activity contexts

Predicted Class								Actual Class
lying	Driving	Running	Jogging	Walking	Sitting	Standing		
1387	2	0	0	3	5	0	Lying	
1	**315**	0	0	0	0	1	Driving	
0	0	**244**	9	3	0	0	Running	
0	0	0	**468**	3	0	0	Jogging	
2	2	0	4	**2197**	8	0	Walking	
21	5	0	0	14	**2125**	1	Sitting	
1	0	1	1	0	0	**261**	Standing	

The confusion matrix further shows that most of the models' prediction errors were due to confusion between *lying* and *sitting*. However, kNN and Random Forest models could sufficiently discriminate between these activities. SMO's produces the highest prediction errors of all the models as can be seen in Table 6.5. Another set of activities that the algorithms could not sufficiently classify are *jogging* and *running.*

They were classified sometimes as *walking,* though the number of cases is very low. Nonetheless, Random Forest and MLP could sufficiently discriminate between these closely similar activities, whereas SVM produced poor accuracy.

The simple reason for the classification error is that the periodic patterns for *walking and running* vary with individuals. The same issue was observed between *sitting* and *standing*. Nevertheless, note that models such as Random Forest, kNN and MLP could recognize these activities accurately. Confusion matrices for other classification models are provided in Appendix B.

6.4.4 Recognizing activity and corresponding location contexts

In the previous experiments, performances of the models were evaluated based on simple contexts (without location). In this experiment, we have evaluated the models' classification accuracies for specific contexts taking into consideration the recognition of location where users perform such activity. The following are the contexts: *Sitting@Home/Office*, *Sitting in a Bus*, *Standing in a Bus*, *Standing in Train*, *Sitting in Train*, *Standing@Home/Office*, *AscendingInAnElevator*, *DescendingInAnElevator*, *AscendingStairs@Home/Office*, and *Descending Stairs@Home/Office*. The fundamental difference in the periodic event patterns, for example, when the user is sitting at home or in a car or bus is one of the reasons to consider the location of these activities in the recognition process. For example, "*sitting in the office"* is different from "*sitting in a bus*" or "*sitting in a train*" because of the variation in their periodic event patterns. However, there is no significant difference between the event patterns of *Sitting@Home* and *Sitting@Office*, as well as between *Standing@Home* and *Standing@Office,* the difference is only in their semantics. Thus, we aggregated these two activity-location instances. Figure 6.8 shows the performance comparison of the activities with and without location information based RMSE evaluation metric.

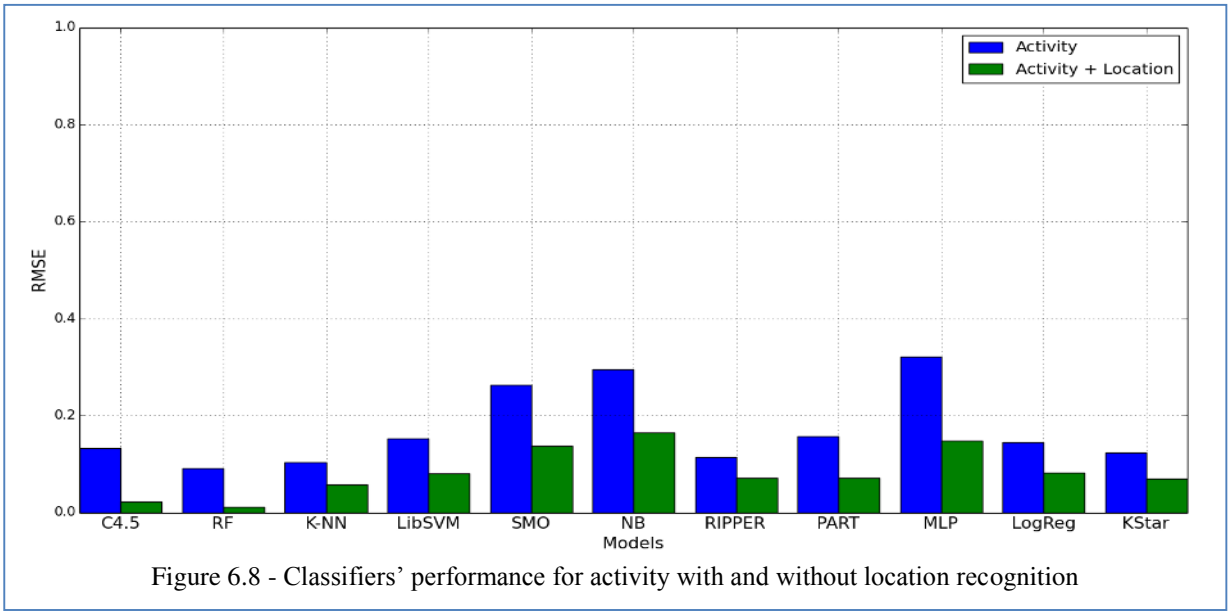


Figure 6.8 - Classifiers' performance for activity with and without location recognition

It is obvious from Figure 6.8 that including location information decreases the recognition error when compared with the results obtained without location information. We observed that SMO, Naïve Bayes and LibSVM have the highest RMSE respectively, whereas kStar, Random Forest, and kNN have the lowest RMSE respectively confirming their better performances. Again, there is an improvement in performance of each model based on recognition accuracy, in particular three algorithms, namely; kStar, kNN and Random Forest produced encouraging performances with accuracies of 86.51%, 90.82% and 91.76% respectively. LibSVM, Naïve Bayes, and SMO produced 64.88%, 57.49%, 71.41% recognition accuracies respectively. One interesting conclusion is that Random Forest performed better than kStar and kNN in context recognition involving locations.

To gain more insights into the number of correctly or incorrectly classified contexts by the models, their confusion matrices for each location/activity pair were computed. Focusing on kNN and SMO models, Tables 6.6 and 6.7 are the confusion matrices for kNN and SMO for recognizing activity with and without location contexts respectively. For other models, please see Appendix B for their confusion matrices.

Table 6.6 – KNN Confusion matrix for location and activity contexts

Predicted Class											
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***	**KNN**	**Actual Class**
2158	5	6	3	1	2	5	6	8	2	*A=Sitting/Home/Office*	
4	**236**	5	0	0	1	5	5	8	8	*B=Standing/Bus*	
4	4	**245**	13	3	7	1	19	2	23	*C=Standing/Train*	
0	0	15	**166**	6	0	3	16	1	5	*D=Ascending/Elevator*	
0	0	2	10	**54**	0	0	7	0	8	*E=Descending/Elevator*	
1	1	4	0	0	**113**	2	0	16	0	*F=DescendingStairs/Home/Office*	
4	6	3	2	1	0	**174**	7	3	1	*G=Sitting/Bus*	
4	6	15	13	2	3	6	**6826**	6	7	*H=Standing/Home/Office*	
5	5	2	1	0	13	1	3	**141**	0	*I=AscendingStairs/Home/Office*	
1	9	24	8	5	0	3	4	5	**240**	***J=Sitting/Train***	

The experiment shows that kNN confuses "*standing in train*" with "*sitting in train*" and "*standing in Home/office*" in 23 and 19 instances respectively. Compared with SMO, "*standing in train*" is confused with "*sitting in train*" and "*standing in Home/office*" in 254 and 104 instances respectively. These confusions contributed to the overall classification errors of this model. Nevertheless, kNN's performance is encouraging with a recognition rate of 90.83%. Considering our data, models based on algorithms such Naïve Bayes performed worse, attaining a recognition rate of 62.37%. It is interesting, however, to observe that Naïve Bayes performed relatively better with more complex activities and locations such as *ascending stairs, descending stairs*, *ascending/descending in an elevator*.

Table 6.7 – SMO Confusion matrix for location and activity context

Predicted Class											
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	SMO	**Actual Class**
2034	9	7	11	1	1	4	119	9	1	*A=Sitting/Home/Office*	
24	**129**	2	2	0	3	21	78	8	5	*B=Standing/Bus*	
3	8	**18**	7	0	0	4	254	4	23	*C=Standing/Train*	
0	7	8	**41**	0	1	1	134	2	18	*D=Ascending/Elevator*	
0	3	3	18	**9**	0	0	41	0	7	*E=Descending/Elevator*	
2	0	0	0	0	**97**	3	19	16	0	*F=DescendingStairs/Home/Office*	
6	17	5	3	0	2	**113**	50	5	0	*G=Sitting/Bus*	
50	13	4	18	4	3	22	**599**	16	15	*H=Standing/Home/Office*	
9	7	0	1	0	25	9	3	**116**	1	*I=AscendingStairs/Home/Office*	
0	14	7	10	0	0	2	104	9	**153**	***J=Sitting/Train***	

6.4.5 Context recognition using rotation vector, orientation and accelerometer data

In the experiments of the last sections, data collected from tri-axial accelerometer sensor were used. In literature, existing studies generally rely on only acceleration data from either single or multiple accelerometers and do not take the orientation of the sensors into account [104]. Bao and Intille [114] argue that using multiple accelerometers can improve the recognition accuracy. Accelerometer generally responds to motion faster, but this comes at a cost, which is noisy data. Smoothing the accelerometer signals to eliminate noise could result in response lags and, that could affect the correct capturing of events. On the other hand, data from sensors, such as a gyroscope and magnetometer or geomagnetic sensors can provide a low noise angle measure [112]. This means that gyroscope data become quickly unrealistic, making the captured data different from the real situation. Combination of accelerometer and other sensors e.g. a gyroscope, a magnetometer or geomagnetic sensor could compensate for the deficiencies. The accelerometer can be used to eliminate the drifts in data from gyroscope and at the same time with gyroscope helping to minimize the inherent noise in accelerometer data. Another reason for combining data from these sensors is to make the context recognition model orientation and rotation independent. Otherwise, this could lead to some erroneous recognition by the models. In literature, existing work use multiple wearable sensors, which are usually attached to specific body positions with specific orientations to achieve better performance. We argue that using data from multiple smartphone sensors could compensate the individual sensor's deficiency.

In this section, rather than using only multiple accelerometers, we evaluated the suitability of recognizing user's contexts using two additional sensors namely, Android based synthetic rotation vector and orientation sensors. To understand the impact of using a combination of these sensors, we compare the performances of the context recognition models, using data from the combination of orientation and rotation vector sensors with accelerometer. Besides, we examined not only the possibility of recognizing user contexts using orientation sensors, but also show the capability of the rotation vector sensor for accurate context recognition. We evaluated the following sensor combinations using recognition accuracies and RMSE as metrics.

a) Accelerometer, orientation, and rotation sensing independently.
b) Accelerometer and orientation sensing combined.
c) Accelerometer and rotation sensing, combined.

The performance comparison of context recognition independently using data from accelerometer, orientation, and rotation vector sensors, as inputs to the recognition models is shown in Figure 6.9. We observed that orientation and rotation vector sensors could be used independently for accurate context recognition. In fact, orientation based context data, performed better than the accelerometer with some models. C4.5, Random Forest, Naïve Bayes, PART, KStar, RIPPER, performs better with orientation data compared with accelerometer. The performance of the rotation vector sensor is also encouraging, competing favorably with accelerometer and orientation sensors. The combination of accelerometer with either orientation or rotation vector sensor could improve recognition. All the recognition models achieved better recognition accuracies using FScore as recognition accuracy measure but SMO and LibSVM produced lower accuracies than the other models. We also evaluated the classifications errors of these models in order to gain more insights into their performance. Figure 6.10 shows the RMSE of all evaluated models showing that combining these sensors with accelerometer could essentially minimize the recognition errors. Nevertheless, LibSVM, SMO and Naïve Bayes produced higher RMSEs than the other models. Another important conclusion to draw from this experiment is that classification models can infer user's activity context independently from the rotation vector sensor.

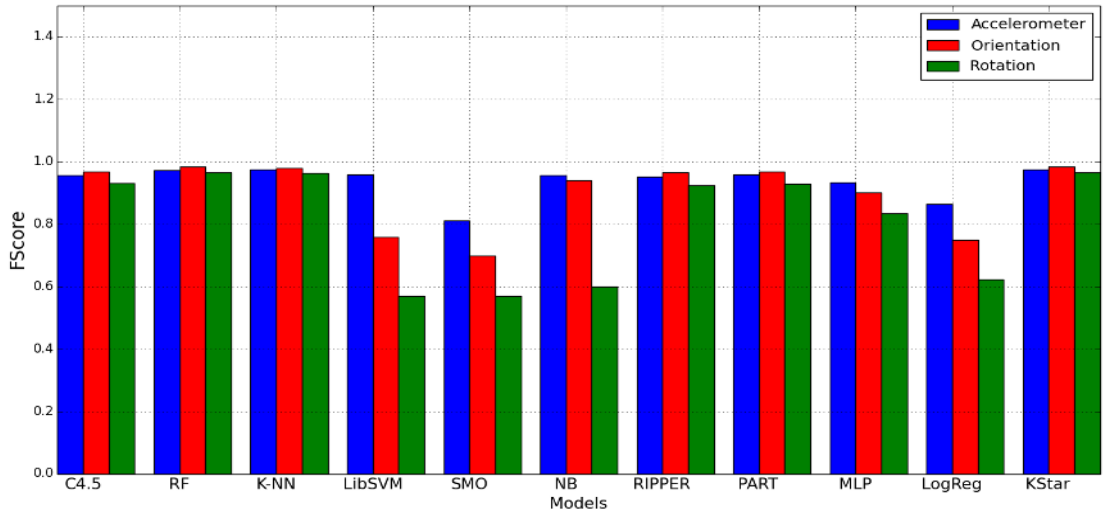


Figure 6.9 - Recognition evaluation using accelerometer, orientation, and rotation vector data independently

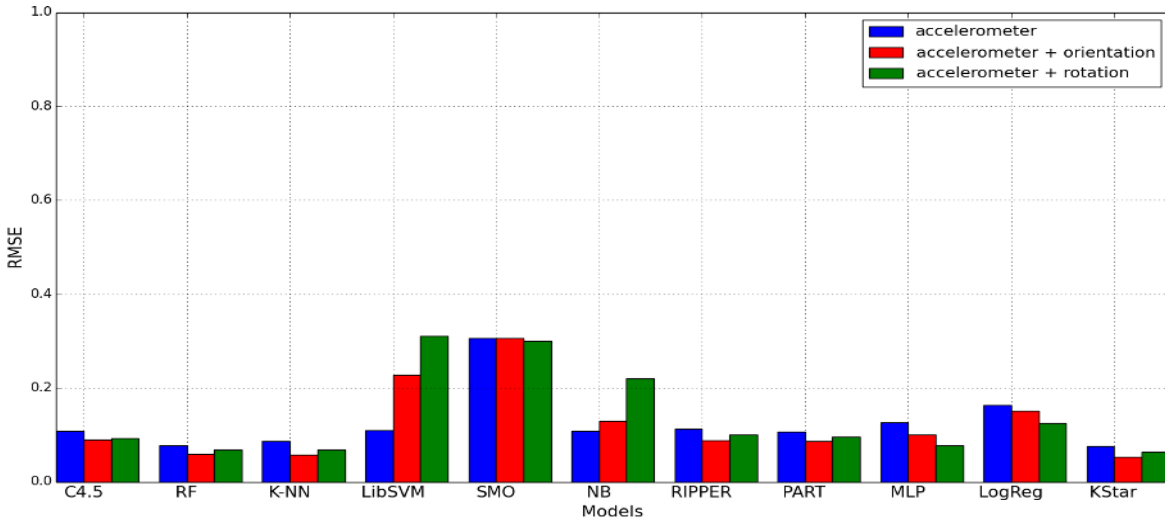


Figure 6.10 - RMSE Comparison for accelerometer, accelerometer + orientation, and accelerometer + rotation vector.

6.4.6 Performance using independent test data

In the previous experiments, the evaluations were performed based on the cross validation approach, which is commonly used in the machine learning literature. However, to obtain an even more reliable result, we have conducted experiments using test data, which were not included in the training dataset. This means that data from all but one user were used as a training dataset, but reserving one as test data. This type of tests allowed us to evaluate the performance of the system under totally new conditions, thus unifying the system response to new data. We designated this type experiments as "user-independent performance evaluation". We collected data from accelerometer and orientation sensors, based on the performance observed from combining these sensors in section 6.4.5. Thus, here we want to evaluate the capability of the models to recognize contextual information of other users other than those whose data have been used to build the recognition model. The F-Score of all the recognition models (except SMO), as shown in Figure 6.11, drops as expected. However, this result generally shows encouraging performance. Recognition models, such as those based on kNN achieved 0.79 accuracy, which falls within an acceptable range compared with existing work [105]. One interesting observation is that other algorithms such as SMO, Naïve Bayes and LogReg performed better than kNN algorithm when completely different user data are used as test data, such as those based on kNN achieved 79% accuracy, which falls within an acceptable range compared with existing works [105].

6.4.7 Real time context recognition evaluation

The performance evaluation executed in this experiment was designed to understand the real-time context recognition capability of the proposed system. This experiment was not designed to evaluate all possible recognition models and contexts in real time. However, based on the results of the offline evaluation of the models, we implemented k-nearest neighbor context recognition model on two devices.

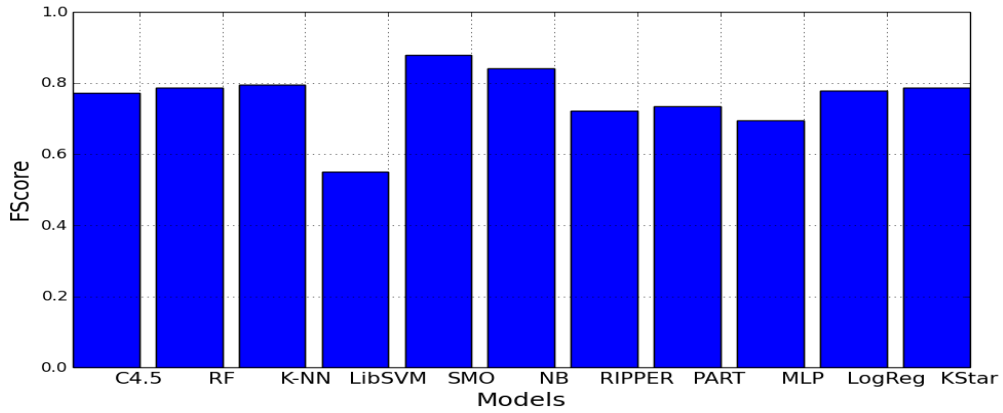


Figure 6.11 - User independent evaluation of classification models for recognition of activities

First is a Galaxy S I9000 smartphone and the other is Galaxy Tab GT –P6200 both running Android OS. The evaluation procedure is straightforward. A test user carries the device around performing specified activities. We then checked the context browser to evaluate if the device reported accurate recognition of the activity context being performed. Each activity was repeated 5 times and the average of the number of times the device correctly recognized the performed activity was taken. Figure 6.12 represents the screenshots during this exercise, while Figure 6.13 illustrates the recognition accuracy for 8 activity contexts, which are :(*L = Laying, R = Running, J=Jogging, W = Walking, S = Sitting,ST = Standing, C=ClimbingStairs, RE = RidingElevator*). The result show that the application is able to recognize these contexts with some acceptable accuracy. For example, 6 of the 8 evaluated activity contexts were recognized with more than 60% accuracy, with some such as *Sitting* achieving more about 73%. Thus, this evaluation show that the recognition model not only recognizes user activities in offline mode, but that it can also recoze these context dynamically(online mode).

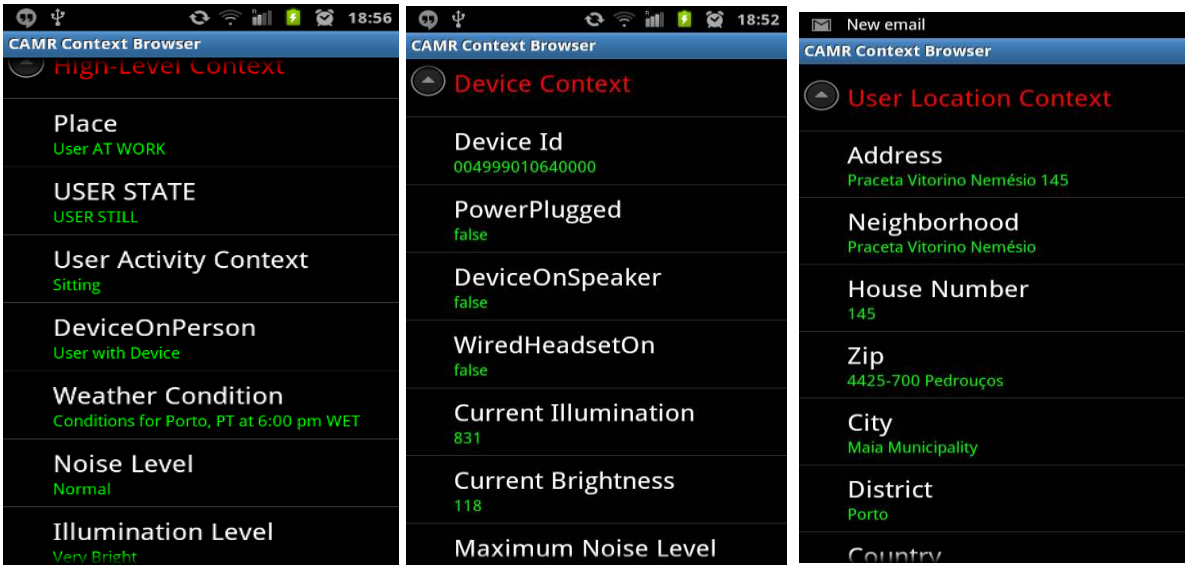


Figure 6.12 - Context browser showing some selected real-time recognized contexts and activities

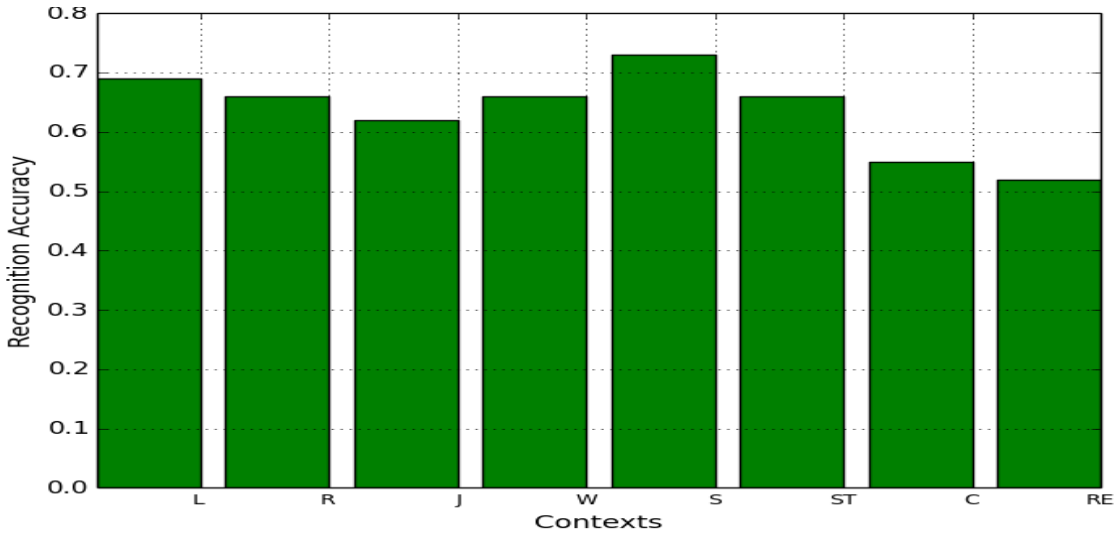


Figure 6.13 -Real-time recognition accuracy of some basic user activity contexts

Table 6.8- Comparison of various activity context recognition accuracy

Reference/Features	Evaluation Method	Sensor	Algorithm	Accuracy
[116] Average, STD, Average Absolute Difference, Time between peaks, bin distribution	10-fold cross validation	Accelerometer	C4.5,MLP,Logistic	>90%
[97] Mean, STD, Entropy, correlation	10-fold cross validation	Accelerometer	Naïve Bayes, KNN,SVM,LibSVM	>90%
[107] Variance, Average, FFT	N/A	Accelerometer, Camera	N/A	>96%
[111]FFT	10-fold cross validation	GPS & Accelerometer	SVM	N/A
[101]Intensity	10-fold cross validation	Accelerometer	MLP,C4.5	99.74%
[115] Sum, SOS, STD, Min, Max, Percentiles	10-fold cross validation & User Independent	Accelerometer	KNN, QDA	94.5%-95.4%
[103]Mean, FFT, STD, Correlation	10-fold cross validation	Accelerometer	BN,NB, SMO, RIPPER,KNN	83%-99%
[118] Mean, Min, Max, STD, Correlation, Zero Cross	10-fold cross validation	Accelerometer & Orientation	Bayesian Network(BN), Naïve Bayes(NB)-NN,MLP,Best First Tree, Decision Table, K-star	>90%
[193] Mean, Max,Min,Variance,Average,FFT,Pearson Correlation	10-fold cross validation & User Independent	Accelerometer & Orientation	KNN,DecisionTrees, SVM	68-99%
CAMR: Mean, Max, Min, STD, Zero Crossing, Median, Range, Sum of Square, Root Mean Square and Variance	10-fold cross validation & User Independent	Accelerometer, Orientation and Rotation Vector	NB, KNN, MLP,KStar, RIPPER, PART, C4.5,Random Forest, Logistic, SMO and LibSVM	>98%

6.4.8 Context Recognition evaluation summary

Generally, the context recognition evaluation shows that it is possible to automate the context recognition process. The results presented in this section show how context information, especially mobile user activity contexts can be identified using data from in-built sensors of user's handheld devices. The advantage of this process is that rather than depending on the users to supply context manually for the recommendation process like most existing work, we can automate this process using the context recognition model. In the next sections, we evaluate how context information can be utilized to provide personalized recommendations. We want to know if contextual information can improve recommendation quality.

Summarily, in comparison with other work, Table 6.8 presents some related work, summarizing their evaluation methods, sensors used, algorithms, extracted features, and the achieved recognition accuracies in comparison with ours.

6.5 Context-aware media recommendation evaluations

Evaluating traditional recommendation systems has been extensively researched [194]-[195]. Generally, the result of any recommendation process is a set provided in decreasing order of relevance to the user's preference. In context-aware recommendation systems, however, this ordered list takes into consideration the context in which such recommendation is provided. Thus, the topmost items should be those that are of utmost interest to the user in the given context. It is much more challenging, however, to evaluate context-aware recommendations in mobile environments because of the lack of standard evaluation framework [1]. In the absence of a standard evaluation framework, we evaluated the proposed solution using the developed proof-of-concept application; a context-aware movie recommendation built using the concepts and design principles presented in chapter 5. The application was used for the experimental evaluation, and for a user study based on movie recommendations. The evaluation was designed to answer questions such as "Which recommendations are best for the user in his present context?" An optimal context-aware recommendation system should provide relevant recommendations according to user's preferences in context. Therefore, since every context-aware recommendation should provide recommendations that would be of interest to users in specific contexts, the main objectives of the empirical evaluations of the proposed context-aware recommendation framework therefore are: (1) To evaluate the recommendation efficacies of the context-aware recommendation processes and compare the results with those of traditional recommendation processes (as baselines). (2) To evaluate the accuracy of the proposed contextual user profiling technique. (3) To evaluate its ability and accuracy to provide recommendation to users (friends) with similar tastes in a similar contextual situation. This experiment was designed to use preferences of users who have been in contexts, which are similar to the context of the current (new) user to provide recommendations. (4) To evaluate the ability of CAMR to address new user or coldstart problem of traditional recommendation system. (5) To evaluate the recommendation time. Recommendation time is an important factor, which could affect user's quality of experience, especially if users initiate the recommendation process themselves. Thus, computing the time difference between initiating and presenting recommendations on the user's device is very crucial in improving user's quality of experience. (6) To evaluate its power consumption. For mobile users, power consumption is a critical factor for adopting new technologies. For example, any application that quickly drains the device battery would definitely have low acceptance rates no matter how appealing the functionality such an application provides.
(7) To evaluate the usefulness and the effectiveness of the proposed solution via a user centered evaluation, using a test panel consisting of 20 individuals. In the following sections, we present the experimental procedures for the above mentioned evaluation objectives, and the dataset, and then present the empirical results of the experiments.

6.5.1 Context-aware recommendation application

To evaluate the CAMR, we developed a proof-of-concept application guided by the concepts and the design presented in chapters four and five, consisting of functions representing the operations offered by CAMR.

The client service was developed on top of the Android platform, running on the Galaxy S I9000 smartphone and Samsung Tab GT P2600. The client profits from a number of sensing APIs available on the Android platform to collect contextual data in a predetermined time interval, which are pre-processed and classified by the recognition model before sending the recognized high-level contextual information to the server. The Android client thus serves as a context service provider. It equally serves as the recommendation client service that initiates recommendation processes, consumes the recommendation services, and presents the recommended items to users.

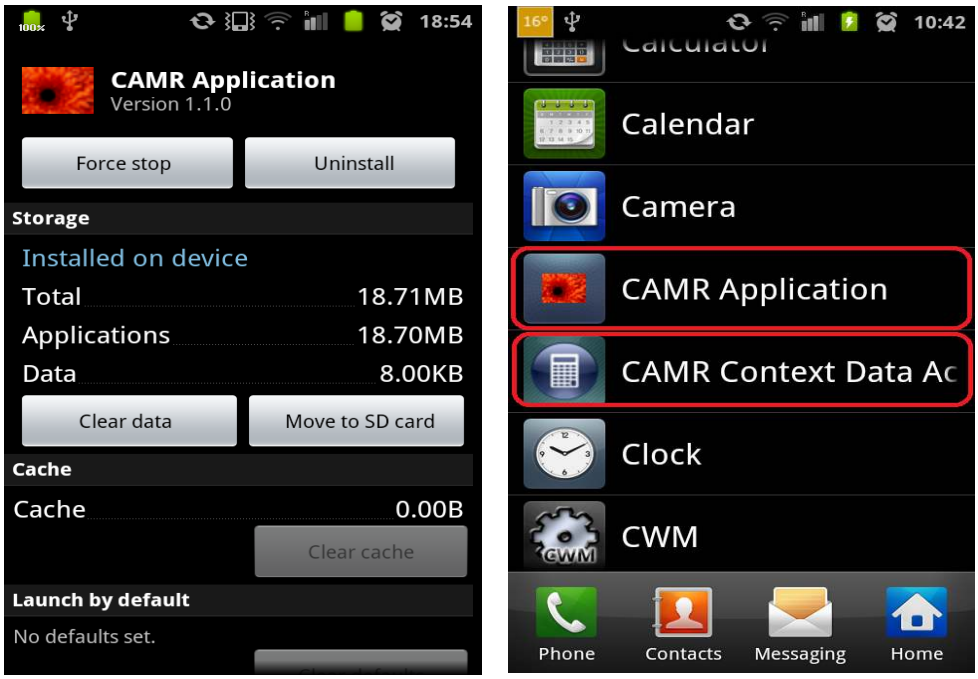


Figure 6.14 CAMR running on Galaxy S I9000 device

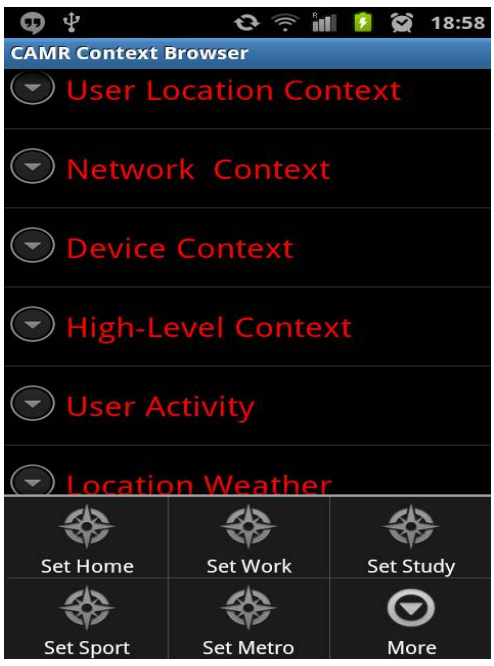


Figure 6.15- User preference setting

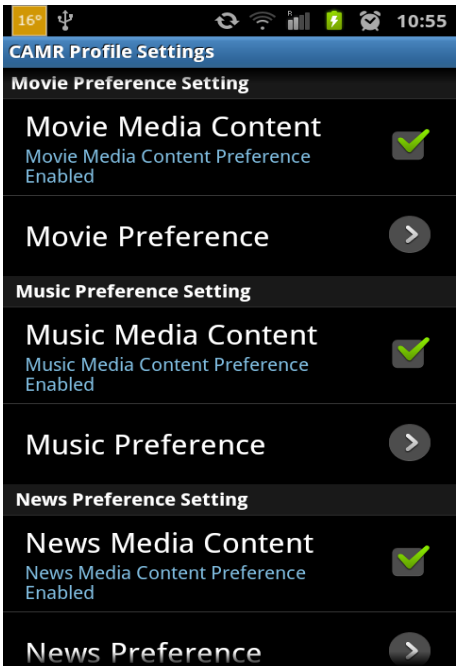


Figure 6.16 - User preference setting

The functionality of the framework was implemented using Java technologies, integrating Java EE 7 (EJB, JPA) and RESTful Web Services, and deployed it on the Oracle Java EE Glassfish server [185]. MySQL 5.6 database serves as the backend, hosting contextual user profiles and media item metadata. Figure 6.14 shows the context-aware mobile recommendation application and its context recognition installed and running on a Galaxy S I9000 smartphone. The contextual user profile service registers user actions and performs all the necessary processing such as the user profiling, update, whereas the recommendation generation is performed by media recommendation service, at the same time working in tandem with the context service and recommendation client service on the Android device. Figure 6.15 shows the user profile management interface, where users can optionally manage their profiles. Figure 6.16 is the context browser where contextual information can be managed on the device. It shows the high-level context indicating, for example, that the user is at location (*Home*), which is obtained using GPS or Wi-Fi, point of interest (POI) and proximity alert. This context information is fed into the contextual user-profiling model, which is stored in a context history repository. Figure 6.17 and Figure 6.18 present some screenshots of the mobile client. For example, Figure 6.17 shows the recommendation interface, which also serves as evaluation interface of the system. It contains the ordered list of recommended item in descending order of contextual relevance. It also contains a checkbox that helps to identify items selected by the user. This information can be used for two things: the evaluation of the recommendations and user feedback collection.

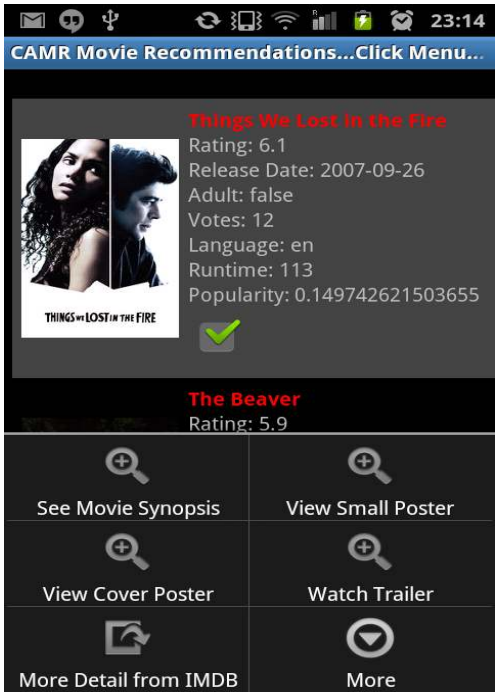


Figure 6.17 - Evaluation interface and recommendation Interface

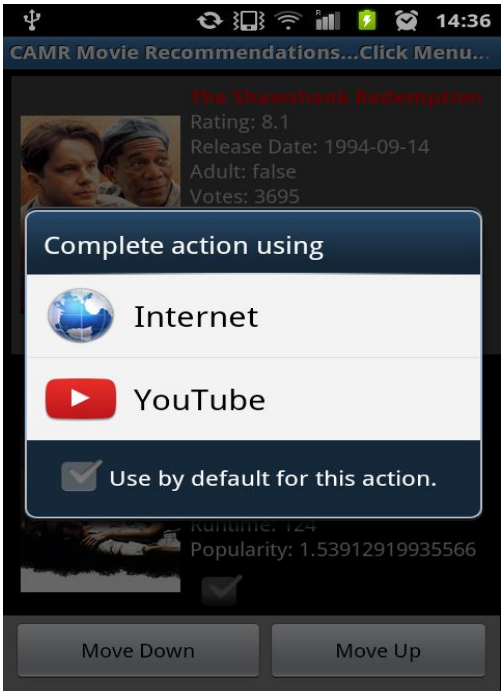


Figure 6.18 - Recommendation presentation interface

Figure 6.19 - Recommendation interface playing recommended movie clip

Figure 6.19 shows the presentation interface for playing the selected items. Additional screenshots of the applications are provided in Appendix A.

6.5.2 Evaluation metrics

This section defines metrics used to evaluate the efficacies and quality of recommendations provided by the proposed framework in controlled laboratory experiments. Evaluation metrics for recommendation systems can be broadly categorized into two types: the predictive accuracy metrics and classification accuracy metrics [194]. The classification metrics unlike the predictive metrics are popular for computing the relevance of recommended items because they do not directly predict ratings of items, but rather measure the ability of the system to provide relevant recommendations.

6.5.2.1 Classification accuracy

Classification accuracy metrics measure the frequencies with which a recommendation system makes correct or incorrect decisions based on whether a recommended item is relevant or not. As illustrated in Figure 6.20 and Figure 21, Precision, Recall, and F-Score are the most popular metrics for measuring classification accuracies of recommendation system. These metrics were borrowed from the information retrieval community because there are standardized metrics for evaluating recommendation systems. It is important to note the difference in the usage of these metrics in both domains. In retrieval systems, such as search engines, the ordering of the returned results is the most important factor. However, in most recommendation systems, this order is usually not considered important; they consider the number of relevant items that are provided in the top n as more important than the order of relevance. In this thesis, the adoption of these metrics to evaluate the proposed framework is based on the importance of the number of relevant items returned by the system as well as the order in which such items are returned.

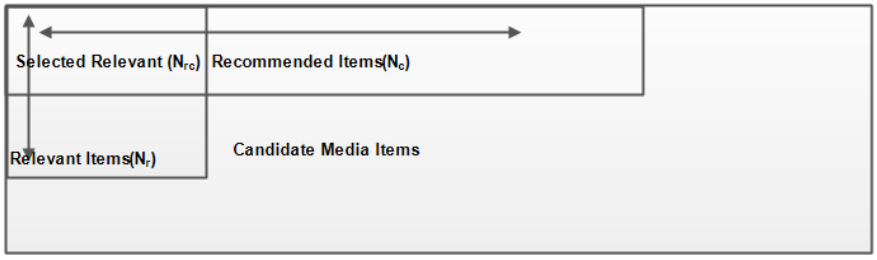


Figure 6.20 - Definition of precision and recall in the context of recommendation systems

(1) **Precision**: Precision can be broadly defined as the proportions of the top recommended items that are relevant. It is the number of relevant items selected from recommended items to the number of items that are relevant in the list. It represents the probability that a selected recommended item is relevant according to the preferences of the user.

In context-aware recommendation systems, precision represents the probability that selected items among those recommended in the current context are relevant in that context as defined in equation (6.5).

$$P = \frac{N_{rc}}{N_C} \tag{6.5}$$

Where N_{rc} is the number of items from the recommendation list selected by the users as relevant in the context of use. N_C. is the number of relevant items.

(2) Recall: Recall is the ratio of the recommended media items relevant and preferred by the users in the current context (for contextual recommendations) to the total number of relevant media items in the recommendation list. In other words, it measures the proportions of all relevant media items included in the top n recommended items.

$$R = \frac{N_{rc}}{N_r} \tag{6.6}$$

Where N_r is the number of selected relevant items in the recommendation set.

(3) F-Score: Is the harmonic mean or weighted average of precision and recall. It is a measure of the accuracy of the recommendation system in the consideration of relevant and non-relevant items included in the recommendation list. In reality, precision and recall are inversely proportional and are both dependent on the number of recommended items returned to the user. When more items are returned, recall tends to increase whilst precision decreases and vice versa. Thus, evaluating the detailed accuracy of the performance of a recommendation system becomes difficult using only precisions and recalls. However, both metrics can be combined using the standard F-Score metric to evaluate the quality of a recommendation system to address this conflict as shown in equation (6.7).

$$F - Score = \frac{2PR}{P + R} \tag{6.7}$$

Example of how to compute these metrics is shown in Figure 6.21, the precision, recall and f-score item number 5 in the ordered recommendation list is obtained as follows.

From the recommendation list, which contains ten items, only seven are relevant.

N_r = 7, N_{rc} = 4, N_c = 5, P = 4/5 = 0.8, R = 4/7 = 0.57, Fscore = (2*0.8 *0.57)/ (0.8+0.57) = 0.67.

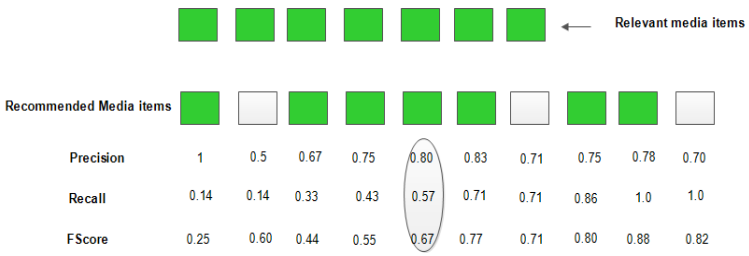


Figure 6.21–Computing precision, recall and f-score for recommendation system

(4) Average Precision (AP@K) is the mean of precision score obtained for each relevant item at top k recommendations on the test cases for every test user computed as follows.

$$\text{AP@K} = 1/m \sum_{i=1}^{m} \frac{1}{k_i} \tag{6.8}$$

Where m is the number of relevant items and $\frac{1}{k_i}$ = 0 if item *i* in the set is not relevant.

(5) Mean Average Precision (MAP@K): MAP is a metric for measuring the accuracy of recommendation systems borrowed from the information retrieval community. It works when a set of recommendation is generated for a group of users with an assumption that we know which of the items in the recommendation set is relevant to each user. This metric can be used to measure the overall recommendation accuracy of the system. Using aggregation over the test users, we compute the Mean Average Precision MAP as the mean of AP@K over all test users as follows:

$$\text{MAP@K} = 1/n \sum_{i=1}^{n} AP_i \tag{6.9}$$

Where n is the total number of test users and APi is the average precision for test user i.

(6) **User satisfaction evaluation metric**: we defined this metric to measure the satisfaction of the users based on the feedback they provide on contextual recommendations they receive.

User satisfaction is computed as follows:

$$S = \frac{\text{No of positive recommendations}}{total\ No\ of\ recommendations} x100\% \tag{6.10}$$

In formula (6.10), positive recommendation is defined as the item considered and selected relevant items by the user.

6.5.2.2 Predictive accuracy

Predictive accuracy metrics are the most widely used recommendation system evaluation metrics. The metrics in this category are borrowed from information retrieval and machine learning communities. The basic assumption for using these metrics is that a recommendation system that accurately predicts rating for an item will most likely provide relevant items that will be preferred by the users. Usually, predictive accuracy metrics are used to evaluate systems that predict ratings (preference) for items that users have never seen, unlike the classification accuracy metrics that are used to measure the relevance of items provided by the recommendation systems. Predictive accuracy metrics measure the difference between rating provided by users to an item recommended to them and the ratings predicted by the system. Examples of this category of metrics are Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Normalized Mean Absolute Error (NMAE). However, because the proposed system was not designed to predict ratings but rather was designed to suggest items that are relevant according to the user's contextual situations, predictive accuracy metrics are not employed in its evaluations. For system such as CAMR, users are only interested in ranked items according to their contextual relevance and not in the ratings of these items. Thus, we consider predictive metrics less appropriate for the context-aware personalization evaluations presented in the next sections.

6.5.3 Experimental design and evaluation

In practice, evaluations of recommendation systems are usually executed in either data driven or user centered experiments [1]. The traditional data driven experiment is usually executed based on dataset containing ratings given to items by users. Using these datasets, behaviors of the users while interacting with the system can be simulated. The basic assumption in data driven experiments is that behaviors of users whose data are used to evaluate the recommendation system is closely similar to the behaviors of the same users when interacting with the system in user centered mode. This method is attractive because it affords the opportunity to evaluate the properties of the systems, which would be practically impossible in the online mode. On the other hand, user-centered involves evaluating the recommendation system with real users while performing recommendation tasks [195]. Although running recommendation experiments in this mode provides a stronger evidence of the system's performance in terms of trustworthiness, user interface or even user intents [194]. However, performing user centered recommendation system experiments is an expensive and time-consuming process. Additionally, it is very complicated to evaluate certain functionality of the system in every possible situation.

Thus, the thesis considers evaluating the contextual recommendation framework first in a data driven process and then executes a simplified user-centered evaluation in a controlled experimental setting. The data driven approach evaluates and compares all context-aware recommendation processes with the baselines, using real life data collected from mobile users. The baselines are the traditional recommendation techniques.

In the user-centered approach, the evaluation was executed using selected contextual information to issue recommendations to users. We then asked the users to provide their level of satisfaction when consuming the recommended items.

6.5.4 Contextual user data acquisition

As earlier said, evaluating contextual recommendation systems is a difficult and expensive task [1], considering issues of privacy, safety, etc. and of course the inability to obtain significant amount of feedback from the users because users are usually not willing to compromise this information [14]. Therefore, carrying out extensive field study using the developed context-aware recommendation system in real life situations is practically difficult. Nevertheless, to evaluate the feasibility of the proposed system, experiments were conducted using context annotated user multimedia consumption data collected from members of the university student community.
These media consumption data and contextual information were collected to build various contextual user profiles in the case study application. In Table 6.9, we show the representative activity contexts characterized by location, time, etc. in which such consumptions were made.

We instantiated over 200 contextual user profiles each having 19 different entries in the genre level (the entry in the category level was the same for all users – movies). These user profiles contain the data, which include contextual preferences of collected from users. The data contain contextual information and multimedia content that the users usually consume in those contexts. Additionally, online movie content metadata obtained by crawling over 4500 movie metadata records from the Movie Database (themoviedb.org), further enhanced with additional metadata retrieved from the IMDB(http://www.imdb.com/). This metadata set contains 23 different movie genres and each record contains on the average three different genre labels. Language, cast, country, duration, and release date characterize the genres. These terms, thus constitute the media item's context in the user profile model as presented in chapter four. First, experiments in sections 6.5.5.1 to 6.5.5.4 are based on the data driven approach while section 6.5.5.5 presents the user centered experimental evaluation.

Table 6.9- Sample contextual information used in the experimental evaluations

User	High-level Contexts
User 1	{(DayOfWeek:Sunday),(TimeOfDay:Evening),(Activity:Sitting),(Location:Cinema), (Illumination: Bright),(Noise Level: Normal)} …
User 2	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location:Home), (Illumination: Bright),(Noise Level: Normal)} ….
User 3	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Jogging),(Location:Sport Complex), (Illumination: Bright), (Noise Level: Normal)}….
User 4	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Sitting),(Location:Office), (Illumination: Bright),(Noise Level: Normal)} ….
User 5 …	{(DayOfWeek:Friday),(TimeOfDay:Evening),(Activity:Sitting),(Location:Home), (Illumination: Bright),(Noise Level: Normal)} …
User n	{(DayOfWeek:Monday),(TimeOfDay:Evening),(Activity:Sitting),(Location:Office), (Illumination: Bright),(Noise Level: Normal)} …

6.5.5 Context-aware multimedia content classification evaluation

The analysis of CAMR framework focuses on the quality of context-aware media recommendation processes rather than on user interaction experiences, e.g. user interface design. Many research works similar to CAMR have carried out this kind of evaluations [26]. As discussed earlier, the main goal of the experiments is to analyze the efficacies of context-aware recommendation processes over traditional baseline approaches to demonstrate the usefulness of contextual information in personalized recommendations. To this end, the deployment of the experimental application was done on separate devices as depicted in Figure 6.22.

The server subsystem was deployed on a Toshiba Satellite laptop with adequate processing capabilities: its processor is Intel® dual core™ i7-3610QMCPU @ 2.3GHz and 2.29 GHz and 8GB ram including a 700GB hard drive, running Windows 8.1 pro(64 bit). The client subsystem was deployed on two mobile devices with broadband Internet connectivity. The two mobile devices include one Samsung Galaxy S I9000 running Android 3.20 and a Galaxy Tab GT-P6200 running Android 4.0 operating systems.

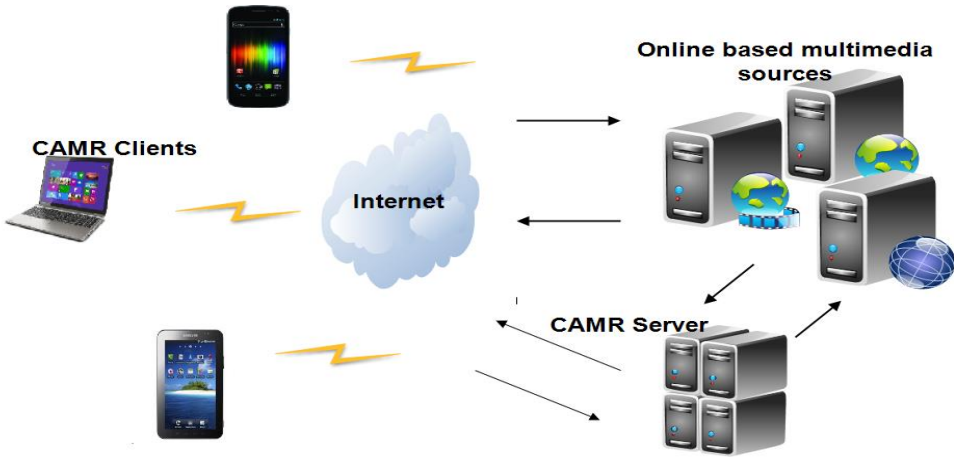


Figure 6.22 - Controlled experimental setup

6.5.5.1 Data driven context-aware recommendation evaluations

The evaluations executed in the following sections involved using the collected user profile data with randomly selected user profiles as target users. These evaluations we designed to enable adequate observations, testing, and evaluation of the feasibility as well as validity of the proposed framework under contextual and non-contextual situations. It also allows us to understand the performance of the behavior of the framework for specific users and for the entire corpus of user profiles. This set of evaluations do not involve collection of feedbacks from real users. The second evaluation involved user profiles with specific content associated to a category-genre-property-context node in the profile as explained in chapter four. This is the evaluation of the contextual recommendation quality.

Experimental setup

To evaluate the quality of the recommendations, given that 200 users whose contextual data were instantiated in the application were anonymous and thus most of them were not available to provide continuous on-device feedback, we devised an approach to allow marking recommended items as *relevant* or as ir*relevant*.

This allows simulating and evaluating the acceptance or rejection of suggested content by the users as previously shown in Figure 6.17. In the evaluation process, an item is marked as relevant provided at least two-third of the terms that appear in its metadata records also appear in the user profile with intensity larger than the average intensity of all terms in the user profile. For example, a suggested movie item with a metadata record presenting three terms is marked as relevant if at least two of those three terms appear in the user profile with intensity larger than the average intensity of all terms (in the *category-genre-property-context* nodes). Otherwise, it is marked as irrelevant. We adopted this approach because we observed in preliminary experiments we conducted that the classification of candidate items is influenced by the number of terms contained in their metadata records, which in the profile are associated with specific contexts, particularly the *genre-property*. Additionally, to compare the processes with the baselines, we conducted the experiments in two parts: experiments evaluating performances of the recommendation processes with contexts, and experiments evaluating their performances without contexts. The first part involved users who have consumed content at specific contextual situations; this is the context-based scenario evaluation. In practice, this translates to using a large variance for setting bigger weights for preferences assigned to genres and properties in contexts, whereas others are assigned smaller weights or even zero since the user does not prefer content with such genres in that specific context.

To evaluate its performance, we computed the precision for five test users, which is the ratio of the number of recommended content in a given context to the amount of content recommended. These users are those who have consumed content in various contexts. Additionally, unlike the evaluations in context-based scenarios, we evaluated situations where content was only associated with the *category-genre-property*, i.e. the context-free version of the processes that do not consider contexts in the personalization process. Like in the context-based scenario, we computed the precision for the selected users and compared the result of both situations to determine if contexts indeed could enhance recommendation quality, i.e., if contexts could assist recommendation systems to deliver more relevant items. Results in Figures 6.23 to 6.26 are the precision plots comparing the contextual and non-contextual content based recommendation processes, for randomly selected 5 target users with *topN* recommendations[6] , n = 5, 10,15,20,25. Figure 6.23 and Figure 6.24 represent the comparison of context-based content recommendations and context-based collaborative recommendations versus the traditional baseline approach, where @a and @b on the x-axis represent the context and baseline recommendations at topN respectively. The results consistently show that contextualization of user profiles provides improved performance over the baseline approaches.

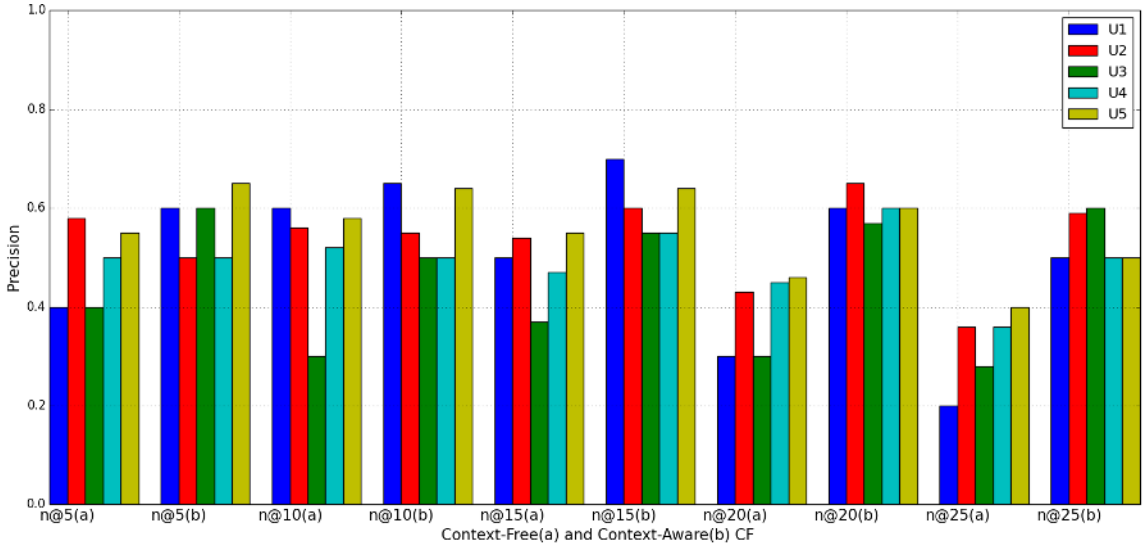


Figure 6.23- Comparison of collaborative (a) and context-aware collaborative (b) based recommendations

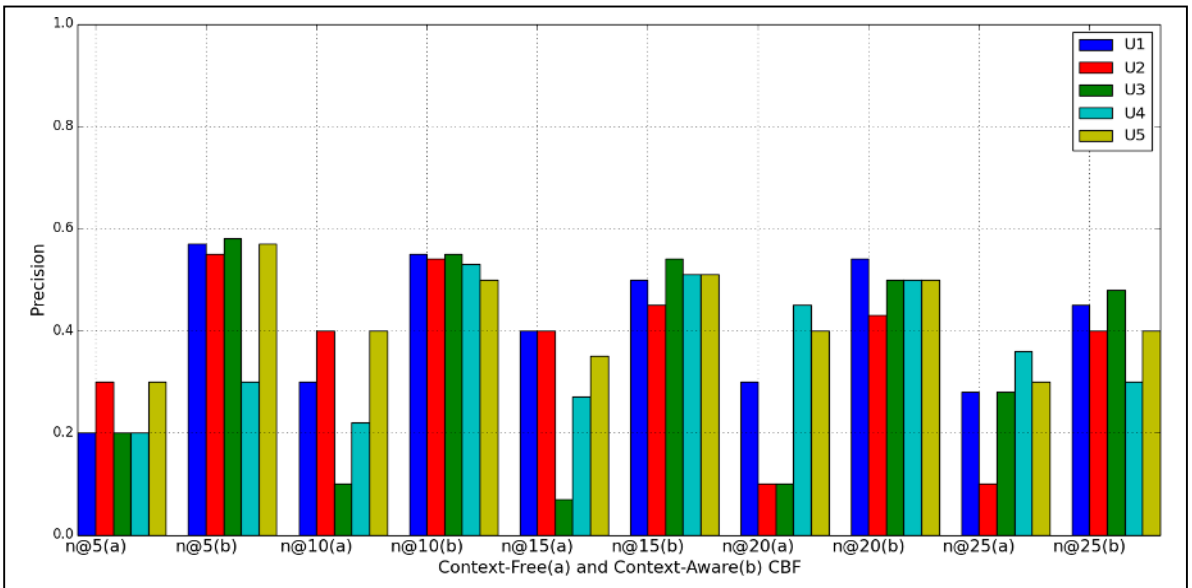


Figure 6.24- Content-based and (a) context-aware content based (b) recommendation precisions

Generally, the reason for the poor performance of the baseline as expected was that in the user profile filtering process, some irrelevant preferences have been included in the process thereby leading to some false positives. On the other hand, the context-based filtering could filter out those irrelevant preferences.

Similarly, in Figure 6.25, context-based hybrid recommendation's performance is compared with the baseline hybrid approach. The result shows some improvement in the recommendation performance.

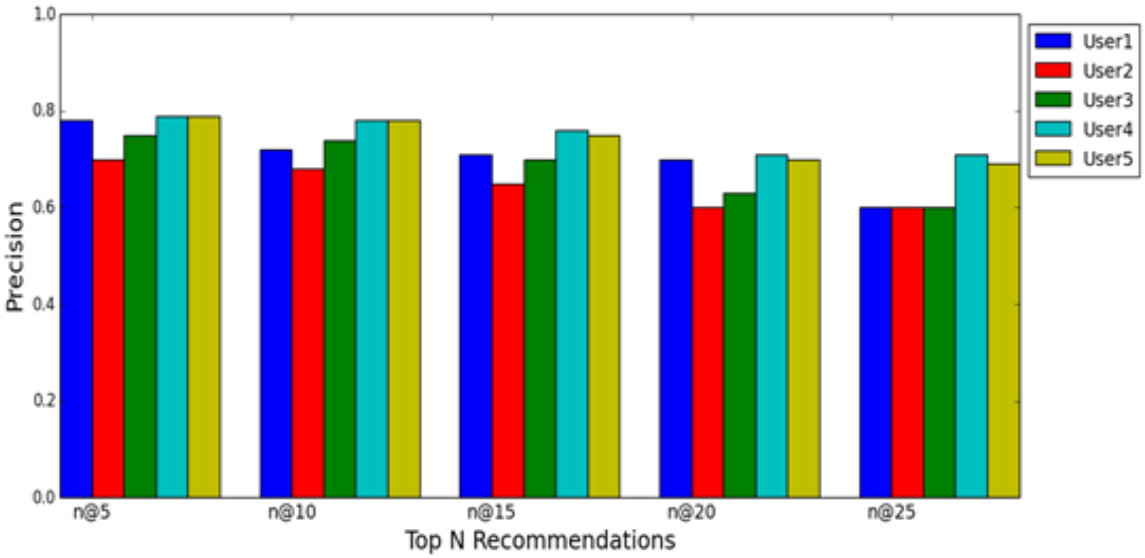


Figure 6.25- Performance combining context-aware hybrid process

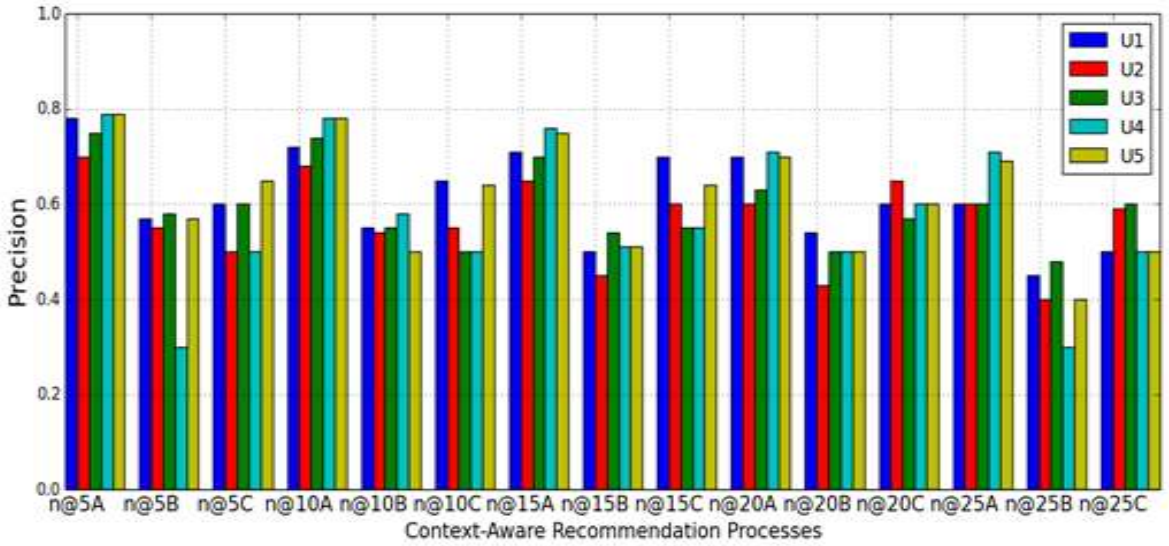


Figure 6.26- Comparisons of context-aware processes (CACR, CACBR and CAHR)

However, Figure 6.26 shows the performance comparison of the three contextual recommendation processes, where n@5A,n@5B, n@5C represent context-based hybrid recommendation (A), context-aware content-based recommendation (B), and context-aware collaborative(C) and so on. The result shows that context based collaborative recommendation performed slightly better than the context-based content recommendations, with context-based hybrid having the best performance.

6.5.5.2 General performance using data driven evaluation

This experiment is similar to the experiment in section 6.5.5.1 except that recommendation quality for all users represented in the contextual user profile repository were computed, unlike in the previous experiments where the computations were based on selecting and focusing on specific number of users as target users. The purpose of this evaluation is to have a general view on the performance of the framework for all users represented in the user profiling process. This way, we would be able to examine the overall feasibility of the framework before conducting user-centered evaluations. Thus, rather than using precision as used in the last section, we used Average Precision (AP), which is computed taking average of the precisions obtained for all users and for all topN recommendations generated. AP provides a single measure of quality across precision levels for multiple users. For a single recommendation generation, the average precision is the set of precision values obtained for a set of top n items provided as recommendation over the number times recommendations are provided. Therefore, for each profile, the average precision was computed for top N (n = 5, 10, 15, 20, 25) using each context-based recommendation process. In our case, this means the set of all relevant recommended items for each user. Figure 6.27 shows the general performance for all users for all the recommendation processes. The results show similar trends as observed when we evaluated specific target users. However, there is a general drop in the recommendation quality compared to experiments involving target users, possibly due to the fact the precisions represent the average for all users and for each top n recommendations since in real life, all items provided cannot not satisfy all users. This issue would be examined further in the user-centered evaluation, where 20 real users evaluated the system via feedback.

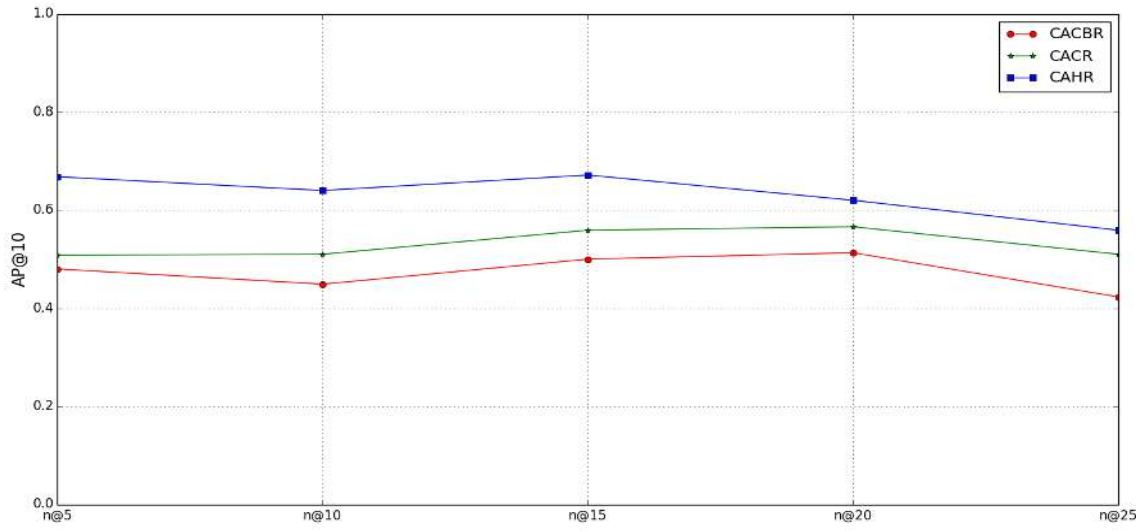


Figure 6.27- Average precisions for all users

6.5.5.3 Performance of context based group recommendations

To understand the ability of the proposed system to recommend items to users who consume similar items in similar context, we evaluated the performance of the recommendation process. The goal of this experiment is to evaluate the benefits of using contextual information to predict what users in similar contextual situation would prefer. The main hypothesis in the group recommendation process is that users who consume items in similar contexts would probably prefer similar items in such contexts in the future. Thus, in this experiment, using the 200 user profiles in the profile repository, these individuals were divided into five groups, each group consisting of approximately 40 members based on the computed similarities between them. A user from each group was then provided with recommendations in selected contextual situations. In addition, recommendation was provided without contexts. Both representing context-aware recommendation and the traditional recommendation methods respectively.

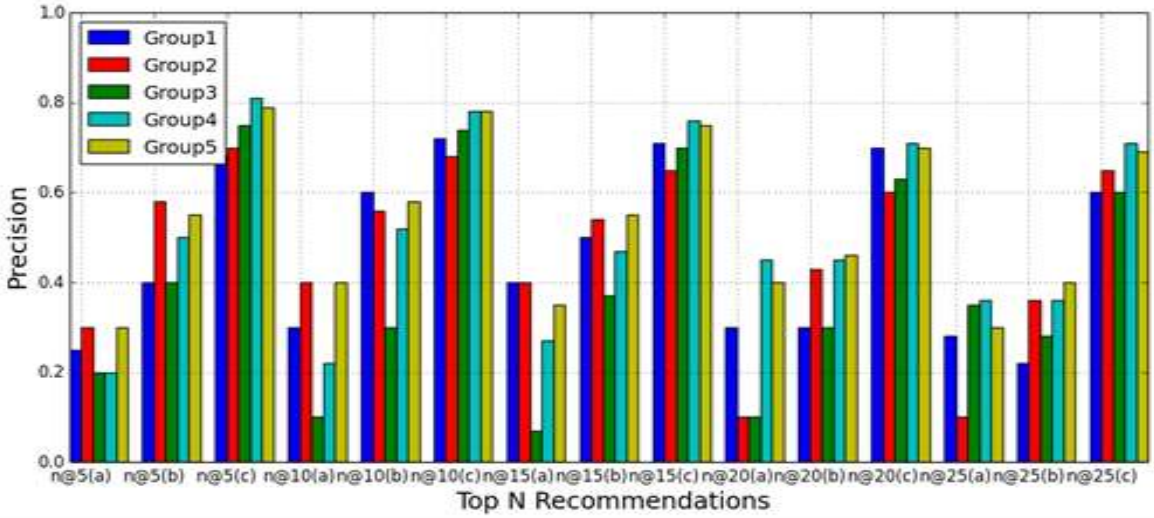


Figure 6.28 - Using contextual information of group of users for content recommendations.

The recommendation was repeated for n = 5,10,15,20, and 25, which represent n items with highest inferred preferences. Figure 6. 28 is the precision plot showing the recommendation quality for each group. Note that a, b, and c represent the precision for contextual content-based, collaborative-based and hybrid recommendation processes respectively.

The result shows that the hybrid process produced better recommendation quality when finding items consumed in similar contexts for groups of users. The significance of this result is that we can use items consumed by users in similar contexts to provide recommendations for a new user. We evaluate this process in series of experiments presented in the next section.

6.5.5.4 Evaluation of new user recommendations

The performance of the proposed system when recommending items to new users was validated under four carefully selected contextual types in another series of experiments. The goal of the test is to evaluate the performance of the system when providing recommendation to new users or those with inadequate information in the system. We evaluated this process for five test users. These five users are those with little or no information in the user profile model. We consider these users as new users. We chose the selected four types of context since it was not possible to evaluate the system under every possible context. We also generated recommendations for these users in no-contextual situation as illustrated in the model in equation (4.2, chapter 4) to compare performances of the model in those situations. The five types of situations including context types and non-context type are:

(A) @homeWeekendNightSitting, (B) @cinemaWeekendNightStanding,

(C) @homeWeekDayNightStanding, (D) @cinemaWeekDayNightSitting

(E) @noContext.

It is obvious from the above context types that each consists of four contextual information. First, we have the location, second, third and fourth are the day of the week, time of the day and activity respectively. Figure 6.29 shows the recommendation AP@10 under each context type, and in no-context situations. We observe from this figure that in those four contextual situations (context types); the system's recommendation AP is better than in no-context situations.

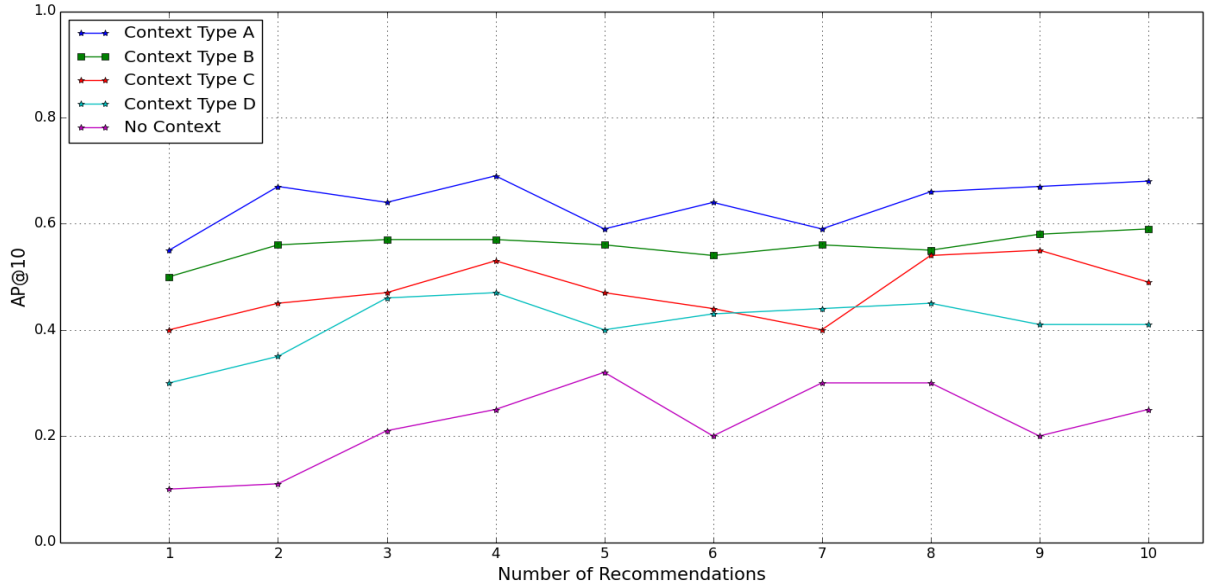


Figure 6.29 - Accuracy of recommendations using a new user's contexts compared to non-contexts

In addition, when we look at the number of recommendations, the APs of the context-based recommendations show clearly that even at lower number of recommendations, the system was able to provide better recommendations than in non-context recommendation. This result shows that the system can provide recommendations for users with little or no information in the system as long as such user's contexts can be identified.

Table 6.10 Mean Average Precision and Confidence Level

(Confidence Interval = 95%) for MAP@10					
Context Type	A	B	C	D	E
MAP@10	0.640	0.559	0.474	0.412	0.224
CL	±0.0287	±0.0155	±0.0318	±0.0308	±0.0446

If we analyze the average precision under each context type, although the recommendations in all of those context types behaved in similar manner, we however observe that *@homeWeekendNightSitting* (A) and *@cinemaWeekendNightStanding* (B) tend to provide better recommendations. The reason is that naturally, and based on the contexts in which users have consumed movie items, it suggests that more neighbors consumed movies in these two context types than in other context types. Table 6.10 shows the statistical confidence level (CL) and the mean average precision at 10 recommendations (MAP@10) for each recommendation provided.

6.5.5.5 User centered evaluation

In addition to evaluating the overall recommendation quality of the proposed CAMR framework, case study scenarios were set up in specific contextual situations in controlled experiments. However, only context-aware hybrid recommendation process was evaluated in those contextual scenarios. The aim of the user-centered evaluation is to provide a generalized performance of the proposed framework involving real users. We evaluated user's satisfaction of the system to ascertain its usefulness and effectiveness to provide understanding of how the framework might perform with actual users.

In the experiment, 20 individuals who were representatives of the system's target users constituted the test panel, whereas 200 user profiles constituted the active contextual profiles in the server subsystem. Considering the difficulty of convincing people to participate voluntarily in this evaluation, we were not able to get all the 200 users whose contextual profiles were incorporated into the system to give feedback. However, judging by the number of users who participated in similar experiments and related work, we have a relatively higher number of participants. For example, evaluation of the system presented by Pessemier et al. in [26], had 16 participants. In [138] and [197], Oku et al. evaluated their systems with 5 and 8 participants respectively. Similarly, in [18] and [196] Yu et al. and Wang et al. evaluated similar systems based on 9 test users with 200 movie records and 10 test users respectively.

The first case study is the home scenario where the user usually performs some activity characterized by other important contexts. These contexts and their values are as follows. *{Location = Home, Activity = Sitting, Day = Friday, Time = Afternoon, Illumination = Bright, Noise level = Normal}* It then provides recommendations to users, based on this contextual information. The second case study is the office scenario where the system has to detect the user's activity and other contexts and use this information to provide recommendations for the users. The detected contexts are *{Location = Office, Activity = Sitting, Day = Monday, Time = Afternoon, Accompanied = No, Illumination = Bright, Noise level = Normal}*. In both scenarios, the users then provide a satisfaction score for each recommendation. For example, in each scenario, 10 recommendations were provided in decreasing order of relevance to the user's contextual situation, using the case study context-aware recommendation application.

We asked the test users to evaluate the degree of satisfaction of the recommendations on a 5-point scale, where "5", the maximum point, means *very satisfied* and "1", the minimum point, means *not satisfied*. We analyzed the feedback provided by these users. We converted the satisfaction scores into binary values as follows.

All scores between 3 and 5 were considered satisfactory; those below 3 were considered not satisfactory. We collected all the scores and then calculated the average to obtain the overall satisfaction from the user's perspective. The overall satisfaction obtained was 69%. Some of the users were excited about the kind of recommendations they received. For example, a test user exclaimed that there is a particular movie she would "*see over and over again*", which happened to be one of recommendations she received. However, some users expressed dissatisfaction with the recommendations, whilst others were indifferent; questioning the need for such an application since they could have easily selected any content they needed, using the content store in their mobile phones or just browse and search for what they need when they need it! Additionally, the final feedbacks from the test users were analyzed and were used to compute precisions and recalls for each test user based on top 10 recommendations. On one hand, the mean average precision (MAP) obtained was 0.65, which shows that the proposed system in real life could provide recommendations that are relevant to each user in specific context with confidence level of 0.65 ± 0.059 (95% confidence interval), thereby minimizing the number of false positives.

On the other hand, the mean average recall (MAR) obtained was 0.56, with confidence level of 0.56 ± 0.029 (95% confidence interval). The AP shows that each user would prefer in specific context about 65% of items in the top in recommendations. The performance shows that there are instances when users would not prefer certain items included in the top n recommendations, depending on the individual's preferences. Note that contextualization per se as used in the CAMR framework is not to provide additional information for the personalized recommendation process but rather, it serves as a means to filter out irrelevant media items that would not be preferred by the user considering the preferences of the user in the present contextual situation. Instances in which the system fails to accurately associate specific contexts to users' preferences probably explain why higher average recalls and precisions were not realized. Additionally, it also confirms that using precision and recall to evaluate contextual and personalized services depend on what is relevant to individuals in specific contexts. Thus, as shown in Figures 6.30 and 6.31, the same average precisions and recalls for individual test users are shown with some achieving better recall and precision than obtained for all users combined. Nonetheless, there is room to improve the overall performance of the system in the future, thereby providing opportunities for further investigations.

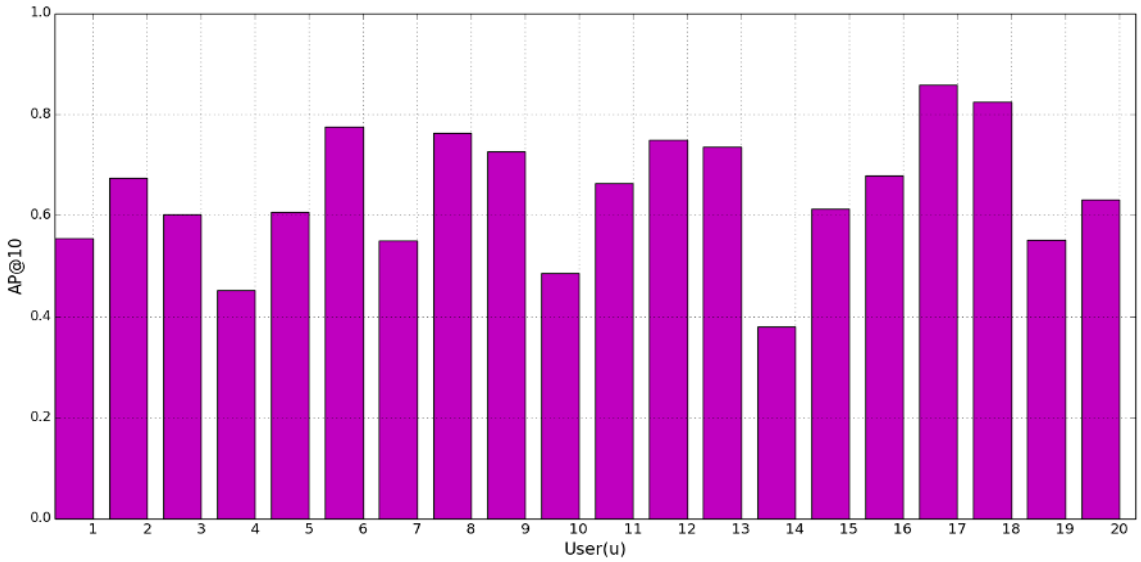


Figure 6.30 - Average precision from test users 'feedback

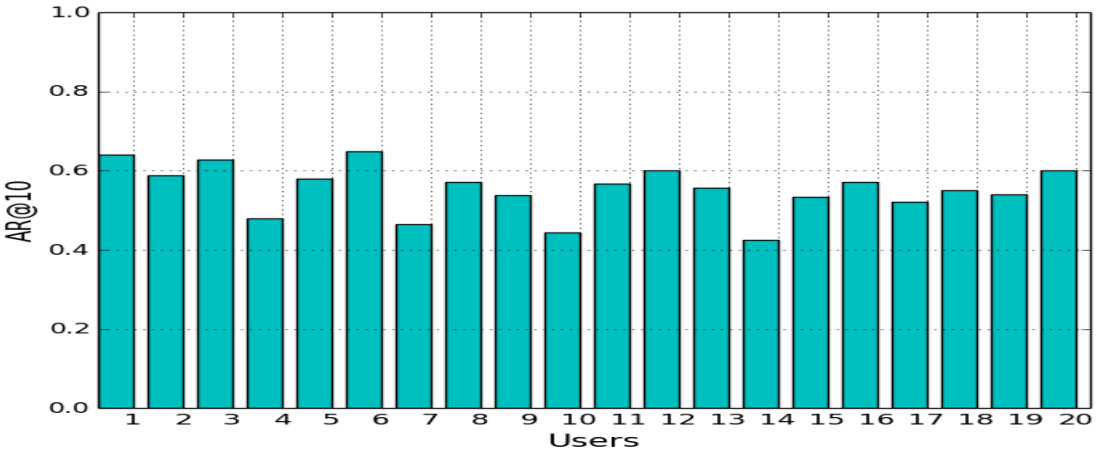


Figure 6.31 - Average recall from test users' feedback

6.5.5.6 Context-aware recommendation application power consumption evaluation

Since CAMR was designed to monitor frequently the device's sensors in order to recognize mobile user's contexts and activities, battery drain becomes a very critical issue. Running GPS, WiFi or cellular network, accelerometer, orientation, rotations, and other sensors have significant impact on the battery lifetime. Therefore, to evaluate the impact of the client service on the battery lifetime, the context recognition service and the recommendation client (both making up the client modules) was deployed on a Samsung Galaxy S I9000 smartphone as the only active application. We then evaluated the system in nine scenarios.

These scenarios and the percentages of power consumed are shown in Table 6.11. The table shows that the smartphone screen consumes a lot of energy in the active state. However, the context recognition service does not need the device screen in an active state to operate at all time. Therefore, the context recognition service was implemented as a background service. This is an important energy optimization consideration in the implementation. In addition, the context recognition service does not have to collect data continuously from all sensors every second because, for example, we can use a motion sensor such as accelerometer or orientation sensor to recognize a user's activity accurately. This observation was established in the experimental evaluations of the accuracies of the context recognition models where we used each motion sensor separately or in combination for context recognition [38]. Therefore, based on this finding and assumption, we implemented another power optimization.

Table 6.11 - Power consumption evaluations of the system

	Application/Sensor	Percentage of Power Consumed
1	All sensors+ Screen	40.66
2	All sensors - Screen	20.84
3	Periodic 30 s + GPS+WiFi+all	8.90
4	Periodic 60 s + GPS+WiFi+all	7.35
5	GPS + WiFi+Accelerometer	5.60
6	GPS+WiFi+Orientation+other	10.78
7	GPS+WiFi+rotation+other	10.40
8	WiFi +other sensors-GPS	7.55
9	GPS +other sensors -WiFi	8.5

The recognition service at every 30 and 60 seconds switched on and off some sensors to conserve power. With screen off, and with all sensors on, the power consumption percentage dropped from 40.66% to 20.84%. In addition, with the periodic switching of sensors, we were able to minimize the power consumption by almost 12% for 30s and about 13% for 60s. Similarly, we optimized the power consumption by switching on and off the GPS sensor, and using either WiFi to sense user location. It was observed that leaving the GPS in active mode resulted in a power consumption surge. It is therefore important to note, from this result, that the proposed system would perform best with battery only when GPS is used as location sensor plus the accelerometer(for motion and activity context recognition) in situations where the user is outdoors, but in situation when the user is indoors, it will use Wi-Fi to conserve battery lifetime. Generally, the result shows that to incorporate more sensors, requires more sophisticated power optimization to prolong the battery lifetime.

6.5.5.7 Recommendation time

In this experiment, we have evaluated the proposed framework using recommendation time. The recommendation time is defined as the time difference between when the framework to initiates a recommendation process and the time when users receive the recommendation set on their devices. This is the time taken to generate recommendations for the current user, starting from when contextual changes trigger the recommendation process. Because in context-aware recommendation, changes in user's contexts triggers the recommendation process, thus we evaluated the time taken to generate a recommendation, starting from when recommendation process is initiated and when an ordered set of recommendations is provided on the client device. One important factor to consider in this experiment is the number of candidate items used for recommendation generation, including the context processing time. The ability of the system to generate recommendations using a large corpus of content in less time is an important feature of any mobile recommendation systems. However, one essential assumption is that time taken to recognize and identify user's contextual information is not taken into consideration. This assumption is because recommendation process is only triggered after context situation of the user has been identified. Context recognition process takes place at interval and runs as a background process on the user device. Therefore, the aim of this experiment is to determine the scalability of the system based on the number of candidate content and time to taken to generate recommendation.

Experiment Setup: We used a repository containing 4500 candidate items, i.e. movie metadata, which were crawled from the Web and filtered into 4500 media profiles. The experiment was then executed in two modes: online and offline modes. In the online mode, candidate media items are filtered during recommendation process. On the other hand, in the offline mode, the media items are pre-filtered i.e. the media profile filtering process is executed before initiating recommendation process. Therefore, when recommendation manager service starts the recommendation process, the contextual user profiling process then filters the user's profile, followed by the generation of contextual user profile and media profile vectors, with subsequent similarity computation and ranking of items according to how relevant they are to the user. The experiment was repeated using 500-4500 candidate items [500,1000,1500,2000,2500,3000,3500,4000,4500]. Figure 6.32 shows the result, recommendation time (ms) against the number of candidate content, to generate online recommendation for a target user. The result shows that in both offline and online operation modes, the recommendation time increases with the number of candidate items increases.

The implication of this is that with large number of candidate items, it would be difficult to sustain the scalability of the system. However, a possible solution is to pre-process the media items, such as executing media profiling process in an offline mode in order to minimize recommendation time, since the result shows that in offline mode, the recommendation time is less compared with the online mode.

The reveals the poor performance of the proposed framework in terms of recommendation time due to computational requirements of the processes. This is because recommendation processes are memory based. This result opens up opportunity to investigate how model based recommendation processes could be incorporated into the CAMR framework proposed by the thesis.

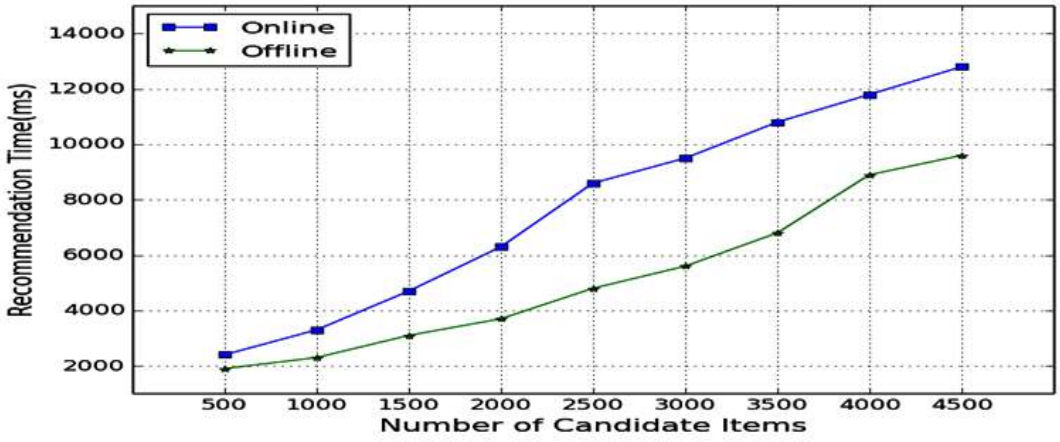


Figure 6.32 - Recommendation time versus number of candidate media items

6.6 Discussion

In this chapter, we have presented experimental evaluations of CAMR's functionality. In the first set of experiments, we evaluated its context awareness capability, using device's built-in sensors to identify mobile user contexts. The experimental results show that good activity context recognition using cross validation [99]-[110] with kNN, Random forest and MLP while LibSVM was found to perform poorly compared to other algorithms. The experimental evaluations also showed that using data from more than one sensor, particularly combining orientation and rotation sensors with accelerometer, could improve the performance of the classifiers.

Although the performance gain was not much, compared to what was reported in [104], encouraging performance improvement ranging from 1.5% to 5% was achieved. One important finding from using these sensors is that instead of using only accelerometer as reported by most previous work [101]-[103] , the experiments established that rotation vector or orientation sensors could be used as independently to recognize user's motion or activity based contexts. The experimental results also established that sliding window lengths play an important role in the overall accuracy of the classification algorithms contrary to existing work [104]. Generally, window lengths of 64 and 128, performed relatively better than larger window lengths of 256,512 and 1024.

In addition, based on the collected context data, the experimental evaluations established that some models could not effectively recognize some specific activity contexts that have similar periodic signal patterns. Some of the models confused contexts such as *Jogging*, *Running*, and *Walking,* probably because signal patterns for these contexts differ significantly, in some cases, for different users depending on how fast they walk or run! Contexts such as *sitting* and *standing* were also wrongly classified. Nonetheless, the algorithms C4.5, kStar, Random forest, kNN and MLP could recognize these contexts using 64 and 128 window lengths with 50% overlap. In addition to recognizing simple contexts, the experimental results also confirmed the capability of the models to recognize activity contexts individually as well as corresponding locations. Three of the models namely, kStar, Random Forest and kNN, all achieved recognition accuracies that are above 90%. The evaluations also established that simple features such as maximum, minimum and range are good enough to recognize most activities using C4.5, Random Forest, kNN, kStar, RIPPER, PART and LibSVM algorithms. The advantage of this process is that rather than depending on the users to supply context manually for the recommendation process like most existing work, this process was automated using the context recognition model.

Generally, context recognition evaluations established the following. (1) That it is possible to use other smartphone sensor other than acceleration to recognize user activities.

(2) That user's activity can be recognized using simple time domain features such as maximum, minimum, range and median. (3) That window length/ size in the feature extraction process can influence activity recognition accuracy. (4) That it is possible to recognize user's activities including the corresponding location where the user performs such activities. (5) We have shown the applicability of the activity recognition process in a mobile multimedia content recommendation application.

In the second set of experiments, we evaluated the efficacies and the effectiveness of context-aware recommendation functionality with promising results. We evaluated the traditional recommendation approaches as baselines to compare their performance with context-based recommendations. Generally, and as expected, the context-based algorithms have better recommendation efficacies than the traditional algorithms [Figures 6.22-6.26]. The traditional recommendation processes showed situations where recommendations without contexts would produce poor recommendation efficacies and vice versa. The rationale is that in the filtering process, user preferences are usually not filtered according to the contexts where users have expressed such preferences. This implies, for example, that content-based recommendations are insensitive to the distributions of user's preferences when contexts where such preferences are expressed are disregarded in the recommendation computations. Therefore, more content that are irrelevant may have been included in the recommendations. Thus, the system uses context information to filter out irrelevant contents, thereby producing better recommendations. For example, if a user prefers watching erotic movies while at home and, if she likes to watch action movies with her friend at the cinema. It is then normal for the system to filter out preferences that relate to watching action movies when the location context is "*cinema*" and the it is "*dry weekend* "or relating to her watching an erotic movie when the location is "*home*" and it is "*wet/raining weekend"*. The hybrid recommendation approach that combines contextual content and collaborative algorithms produced the overall best results, with significant improvement in the recommendation quality. In addition, the contextual hybrid recommendations could be used to minimize new user problems because as the experiment on using similar contexts to group user has established, users in similar contexts tend to consume similar items. As observed in Figure 6.28, users in similar contexts are most likely going to like or consume similar content. For example, in real life, it is normal when people gather at the same location common goal or taste!

Another observation was that contextual hybrid recommendations produced better performance than context-aware content-based or context-aware collaborative recommendation process. This is probably because the system has more information, especially contextual information that better help to filter out irrelevant preferences. In essence, this means that combining context of a target user with those of like-minded users is a better way to personalize recommendations. It is also important to note that context-aware content-based recommendations performed below par compared to context-aware collaborative and context-aware hybrid recommendations. The context-aware content-based recommendations are slightly insensitive to the distribution of contextual preferences in the user profiles. On the other hand, as the number of contextual preferences common with other users increases, the context-aware aware recommendations provided improved recommendation quality and vice versa. The context-aware hybrid recommendations have better performance than the other methods. The context-aware hybrid method, as revealed by the experiments, confirmed that it is more effective than pure context-aware recommendations.

In addition, evaluation under carefully selected contextual situations showed consistently improved AP of recommendations. For example, evaluation of recommendations under no context condition showed poor performance, reflecting the coldstart problem. However, when we compared results of recommendations under contextual situations with those under no context, the contextual recommendations showed clearly, how context can be explored to address coldstart problem, which is obvious in the result as contextual recommendations produced AP of up to 70% compared to that of no context that produced AP ranging between 10% and 29%.

Additionally, the satisfaction obtained from user centered evaluation shows that the proposed context-aware system is feasible in practice. The results obtained from user-centered evaluation, though not as good as those obtained in the data driven experiments, provide some evidence that users are satisfied with the provided recommendations. In terms of power consumption, CAMR performed fairly well, though the performance leaves room for further improvement

via sophisticated power consumption optimization. Nevertheless, the experiments established that running the recognition service continuously with all sensors in active state would drain the battery quicker than if it was left to switch off and on some sensors at intervals. For example, the experimental evaluations also established that continuous monitoring of the GPS is a key power consumer. Therefore, the context recognition was redesigned to monitor Wi-Fi or Cell-ID as the primary source of location information, but to use GPS only when user is outdoors. One of the benefits of the proposed framework is that users do not have to deploy additional wired devices, because it uses the mobile device's built-in sensing resources unlike similar work [106], which did not implement recognition model on the user's mobile device. Generally, compared with some similar existing work, such as those reported by Yu et al. [18] and Pessemier et al. [26], we have been able to establish the performance of the proposed system in terms of recommendation precisions for the three recommendation approaches implemented as Web services. The precisions demonstrate the superior efficacies of context-aware recommendations to traditional recommendations. However, the evaluation of the recommendation time as a measure of its scalability property shows that the proposed framework is computationally expensive.

6.7 Summary

In this chapter, the thesis presents empirical results of experimental evaluations of CAMR via a proof-of-concept context-aware movie recommendation application. The experiments were categorized broadly into two sets: The experimental evaluations of the framework context recognition capability and the efficacies of context-aware recommendation processes. The experiments on context recognition capability show that mobile user's devices can provide accurate contextual information. This improvement can be achieved by investigating more sophisticated power optimization techniques. The results of the experiments on the efficacies of the recommendation processes are effective even though, there is much room to improve the system especially by incorporating more advanced recommendation algorithms. Examples of such methods are context-aware recommendations based on matrix factorization e.g. Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). SVD is a well-established method in information retrieval community [63], [198].

The results show the importance of contextual information for determining the relevance of media items to mobile users. The results show how contextual information can influence the recommendation efficacies, helping to filter out preferences that are not relevant to the target user in the present context during filtering process. One distinctive feature of the proposed system is its ability to combine dynamic context recognition and recommendation processes, unified by the contextual user profile model. The contextual user profile model has a wider scope; it can be extended to serve context-aware filtering processes, using user's contextual preferences. The context service relies on the context recognition model to glean accurate context information from mobile device built-in sensors, using machine-learning technique. It was designed to profit from mobile devices' built-in sensors to enhance seamless and automatic recognition of user's context information without using additional wired devices. The recommendation processes can filter out irrelevant recommendations using user's current context, by concentrating on user preferences that are relevant in the present context, unlike the traditional algorithms that use both relevant and irrelevant preferences for content classification. The user satisfaction evaluation demonstrates that the proposed solution can provide users with relevant items. The user satisfaction experimentation would profit from a well-designed field study, allowing evaluating the performance of the system by users, in different real and natural contextual situations. However, this is an expensive process involving a lot of financial commitment and time on the part of the investigators and test users respectively. Another major weakness of the proposed framework is that it relies on Wi-Fi connectivity between the smartphones and the server. We would like to experiment with mobile networks, such as cellular network.

Chapter 7
Conclusions and Future Work

The emergence of pervasive computing, rapid advancements in broadband and mobile networks and the incredible appeals of smartphones are driving an unprecedented universal access to online media resources. As more and more media services continue to flood the Web, search and retrieval of useful content will become more and more challenging, consuming invaluable time. To offer richer experiences to mobile users, media services must deliver better, faster and transparent access to content of interest in a particular instant, at a particular place and under particular contextual conditions. We argue that such systems will be able to deliver optimal results when equipped with contextual knowledge of the consumption environment as well as contextual preferences of mobile users. This information will not only enable automatic provisioning of personalized multimedia recommendations tailored to the mobile user's contextual preferences but will also facilitate anywhere anytime multimedia access, removing barriers to the adoption of mobile devices as the primary platform for multimedia entertainment. This thesis set out to demonstrate this hypothesis by investigating and developing solution for context-aware recommendation systems supporting the delivery of personalized multimedia content to mobile users. The proposed framework implements recommendation services that rely on the use of a dynamic context sensing, inference and recognition service, which enables the system to build contextualized user profiles. The recommendation engine uses such profiles to provide personalized recommendations. The context recognition service identifies the user's dynamic contextual situation, by collecting data from the mobile device's built-in sensors, extracting relevant features from them, applying reasoning and inference techniques to these sensed data. The user's contextual situation is defined in terms of high-level concepts such as the physical activity (e.g., running or standing) or location (@home or @Office). This chapter draws some conclusions from the investigation and development work conducted in the course of this thesis. Whilst clearly demonstrating the benefits of the proposed solution, it also presents its weaknesses, indicating how these weaknesses and other interesting areas not considered in the thesis can be addressed in the future.

7.1 Conclusions and contributions

Existing solutions to enable the universal access to multimedia content have considered only the challenges of delivering explicitly requested content, taking into consideration a limited set of contextual constraints. They generally overlooked aspects related to anticipating user's needs [179]. Anticipating mobile user's content needs requires an understanding of the user's environmental situations and contexts, which influence their preferences. Systems that combine knowledge of the user's preferences and needs with the knowledge of the user's contexts, such as the activity the user performs or his/her location, thus fusing context-awareness with traditional content classification and recommendation methods, are still needed to address the mobile multimedia content overload problem. The information retrieval and recommendation system communities have researched the information overload problem extensively for desktop-based users using innovative techniques. However, in mobile environments, users engage in different activities compared to those of traditional desktop users and hence have frequently changing preferences as their activities and contextual situations change. Automatic provisioning of personalized multimedia content via personalized recommendations, tailored to user's needs or preferences, based on his contextual situations, will not only facilitate universal access to media content anywhere, anytime, but will also minimize the barriers facing universal adoption of multimedia services on mobile devices now and in the future.

This thesis has identified existing challenges associated with media recommendation systems in mobile environments and consequently has investigated approaches to overcome them.

It argued that, in the first instance, those problems were because existing systems did not consider that the preferences of users are likely to change as their contextual situations change. They were developed with the assumption that if a user liked a media item in the past, she would also like similar items in the future. Alternatively, that the user would probably like the same items consumed by her friends or by other users with similar profiles. However, in mobile environments, these assumptions do not always hold. Mobile users have different preferences in different situations, and in fact, media items liked in the past by friends of the user might not be of interest considering his/her current contextual conditions. For example, a user who likes to hike during weekends might not hike on weekends if there is a downpour. Essentially, mobile user's preferences and tastes are defined by their contextual situations, which are not static.

Because users' contextual situations are not static, but change as they move from one place to another performing different activities, it is important to capture these changes accordingly and incorporate them into the personalized recommendation processes. Existing media recommendation systems that considered contextual information in the recommendation processes relied on static information. Some other systems relied on asking users to provide this information. Using static or manually acquired contextual information is not a desired solution for context-aware personalized recommendations in mobile environments. This thesis investigates how contextual information can be automatically acquired, and how it can be incorporated into personalized recommendation processes.

To address these problems, the thesis has achieved the following results:

First, it proposed and realized dynamic context and activity recognition framework [8] [38]. The thesis developed a context awareness framework that supports dynamic contextual user profile model for adaptive multimedia content classification and recommendations. The developed context awareness framework can execute on the user's device as a service, taking advantage of user device's built-in sensors and classification algorithms to identify real-time mobile user's context information and activities. The context recognition gathers low-level events from the device's built-in sensors, such as accelerometer, orientation, location (GPS, Wi-Fi), light, orientation sensors etc., and preprocesses them to infer high-level contextual information. The high-level contextual information produced in the process can be accessed from any platform via Web services.

Second, it realized a contextualized user profile process that unifies the context awareness framework and multimedia content filtering processes [179] [199] [200]. The context-sensitive user profile model supports and contextualizes recommendation techniques such as content-based filtering, collaborative filtering, and a hybrid process. The contextual user profile model summarizes the user's consumptions into a limited set of categories. Categories are described by a genre; a number of properties in turn describe the genre. Several genres may be associated with each category. Several properties are associated to every genre. This structure thus provides sufficient flexibility to enable the system to provide at different levels, (items of a certain category-genre-property up to items of a certain category only), either using or not the contextual information. To retrieve and to learn the contextual user preference in the contextual profile model, it adopts a case based reasoning (CBR) methodology based on a nearest neighbor algorithm to process and infer user's preferences in the current contextual situation.

Third, the thesis realized the development of a context-aware classification and recommendation framework [199], [200]. The traditional recommendation techniques (collaborative recommendation and content-based filtering) suggest multimedia content to a target user based on the opinions and preferences of other likeminded users, or based on the user's consumption history [12]. These solutions are difficult to apply directly in mobile environments. Thus, the thesis proposes a conceptual context-aware media recommendation and classification framework (CAMR) incorporating contextualized user profile model and context awareness framework in a flexible way to allow incorporating contextual information into existing personalized recommendation processes.

This produced a context-aware content based, context-aware collaborative, and a context-aware content-based collaborative filtering techniques for content classifications and recommendations.

Fourth, the thesis designed and developed a system for context-aware personalized multimedia recommendations [201]. The developed system supports the provisioning of contextual information, contextual user profile as well as recommendation processes as web based services that can interoperate with other services both internally and externally.

For example, there are diverse mobile platforms, which are presently available to mobile users and this keep growing. Thus, designing and developing context-aware personalized recommendations for mobile users requires that such systems operate on different mobile devices, in terms of operating and hardware platforms. Additionally, to demonstrate the feasibility of the proposed solution, the thesis developed an experimental context-aware movie recommendation application [179], [200] based on CAMR framework. The context-aware recommendation application was developed based on the CAMR conceptual framework as a proof-of-concept. This is a concrete implementation of the case study to demonstrate the feasibility and the efficacy of the proposed solution. The context recognition framework, contextual user profile, and context-aware recommendation algorithms are evaluated via the experimental application. This implementation served as a proof-of-concept, and as a platform to evaluate the proposed framework. First, it was used to simulate the case study for evaluation purposes. Second, it was used in a user study to evaluate user satisfaction of the contextual recommendations.

Besides, the thesis proposed the incorporation of an adaptation service to tailor the presentation of the recommended content according to contextual constraints, namely, device characteristics and network conditions [202]. Since most Web-based contents are not specifically designed for consumption in mobile devices, it may happen that the format of the content selected by the user from the list of recommended items, does not match the device's capabilities (e.g., codec not installed or low level of battery left) and/or network conditions (e.g., bandwidth availability) . The thesis, therefore, proposed the incorporation of a knowledge-based solution, which uses ontology and semantic web rule language (SWRL) to determine the need for adapting the recommended content according to the constraints imposed by the device and by the network connection. However, the development of the adaptation decision mechanism and adaptation operations are beyond the scope of the thesis, relying instead on the existing adaptation service developed by Andrade et al. [127].

The proposed framework has been experimentally evaluated via the developed context-aware recommendation application. These experimental evaluations show that personalized delivery of recommendations will profit, in terms of quality, from incorporating contextual information in personalization the process in mobile environments. The results further demonstrate that the proposed context-aware personalized recommendation framework can provide better recommendation efficacy than the traditional recommendations. Its ability to mitigate the new user (coldstart) problem was also validated. A user experience survey of the recommendations provided by the system has been assessed as effective, and the results demonstrate that users are generally pleased with the provided contextual media recommendations. This thesis has demonstrated the effectiveness and efficacy of contextual information in the provisioning of media recommendations to mobile users. The generic nature of the proposed framework makes it applicable in other domains, especially in news, music and other categories of media contents.

In addition, evaluations of the underlying context recognition show its capability for dynamic provisioning of mobile user’s contextual information. Although the evaluation of the developed prototype system demonstrates promising results, both in terms of recommendation performance as well as in terms of power consumption, based on the obtained results, we conclude that more efforts at both the research and optimization levels would be required to bring the system into a product for commercial development.

In particular, there is need for improvement of the proposed solution considering the power consumption and other potential drawbacks that are discussed in the next section.

7.2 Areas of future work

Whilst delivering satisfactory and promising results, the solution proposed by this thesis still presents some weaknesses, thus leaving room for future research and improvements. Some of these weaknesses are discussed in this section, which could be seen as directions for future work. This section also suggests other aspects for further investigation, and alternative approaches that could be experimented to evaluate possible benefits concerning the system's performance.

First, the current system does not provide a means to infer user's contextual situations (at higher semantic level) on the device, though it provides a classification model, which effectively can recognize the user's contexts from low-level sensor data on the device. The inference of higher-level contextual information is done on the server side via the inference model based on ontology and inference rules. It would be of interest to execute this functionality on mobile devices. This will engender further trustworthiness of the system, as well as reducing the latency when computing recommendation but, because of the computational requirement of such implementation, it is better to execute such implementation on the server. Additionally, mobile users' moods are examples of contexts that influence their contextual preferences. However, in this thesis, sensing user's mood and incorporating it into the contextual user profile model was not investigated. We would like to explore user's mood to provide multimedia content recommendations.

Second, the contextual user profile model infers users' preferences in the current contextual situation using a k-nearest neighbor based method. Nearest neighbor method as used in this thesis has been adjudged as one of the fastest and most accurate for providing recommendations. However, based on the advice of peers, we would like to experiment with other methods to explore the possibility of incorporating them in the CAMR framework. For example, the prediction of user preferences in their contextual situations could have alternatively been modeled as a Multi Attribute Decision Making (MADM) problem. Future direction in this regard would explore more of MADM based solutions, which have been proven as efficient in dynamic environments [176], [203]. We would also like to explore other classification or clustering based methods and to compare their performances. This is also applicable to the recommendation processes. The thesis has used the proposed contextual user profile model to extend nearest neighbor-based algorithms for all the recommendation processes implemented. In the future, we plan to explore how to incorporate other algorithms such as singular value decomposition, probabilistic latent semantic analysis, clustering based algorithms, linear regression, etc. for media recommendations in mobile environments, especially to address the scalability difficulties encountered. Additionally, we would like to explore field-based user evaluations, if the resources are available, based on the new recommendation processes in other multimedia domains other than movies, e.g. news, music, etc.

Third, issues of safety, trustworthiness, and privacy have to be investigated. To ensure the present system does not keep user names and other private and sensitive data, we adopted anonymity of users, so that the server has little knowledge of who these users are in terms of names, sex, etc. [47]. This information is provided as settings on the user device. In the recommendation process, the system anonymizes the user profile information. Despite this effort, users have concerns over sharing of their sensitive information, such as location. In fact, this is one of the challenges faced in the course of executing this research, especially during evaluation stages. Some users were reluctant to participate in the data acquisition and in the evaluation processes due to privacy concerns and lack of incentives. In the future, we intend to incorporate more robust and secured interactions in the proposed system. Is it possible to implement the entire CAMR system on the device or using peer-to-peer architecture without relying on client-server architecture or is it better to offload the more computationally expensive processes on the cloud for example? In case of the former, every

processing involving the current user will be executed on the device. If this is possible, will it be possible to implement collaborative-based context-aware recommendation based on this architecture? In the latter case, how would the confidence and trust of users be secured knowing that their data are being used by the system they do not completely control? These are important questions to answer in the future.

Finally, power consumption is a critical issue as far as mobile users are concerned. Battery lifetime of devices remains a concern compared with desktops and as such, any application that consumes much power, no matter how helpful or appealing, would not enjoy mobile user's acceptance. The power consumption evaluation of the proposed system revealed that there is a need to minimize its present power consumption by employing more sophisticated but efficient power optimization techniques.

Bibliography

[1] D. Vallet , M. Fernandez , P. Castells , P. Mylonas and Y. Avrithis. Personalized information retrieval in context. 3rd Int.Workshop on Modeling Retrieval Context 21st Nat. Conf. Artif. Intell., 2007.

[2] J. H. Roh and S. Jin. Personalized advertisement recommendation system based on user profile in the smart phone. In Advanced Communication Technology (ICACT), 2012 14th International Conference On, pp. 1300-1303, 2012.

[3] F. Ricci, L. Roback & B. Shapira. Introduction to recommender systems. In F. Ricci et al. (eds.), Recommender Systems Handbook, 1DOI 10.1007/978-0-387-85820-3_1, © Springer Science+Business Media, LLC, 2011.

[4] F. Xia, N.Y. Asabere, A.M. Ahmed,J. Li, X. Kong. Mobile Multimedia Recommendation in Smart Communities: A Survey. Access, IEEE, vol.1, no., pp.606, 624, 2013.

[5] Z. Yujie and W. Licai. Some challenges for context-aware recommender systems. Computer Science and Education (ICCSE), 2010 5th International Conference on, vol., no., pp.362, 365, 24-27, Aug. 2010.

[6] A. Vetro. MPEG-21 Digital Item Adaptation: Enabling Universal Multimedia Access. IEEE Multimedia Vol.11, No1, pp. 84- 87, 2004.

[7] S. Wong, V. Raghavan. Vector Space Model for Information Retrieval – A Reevaluation. In Proceedings of SIGIR, 167–185, 1994.

[8] A.M. Otebolaku and M.T. Andrade. Context Representation for Context-Aware Mobile Multimedia Recommendation. In Proceedings of the 15th IASTED International Conference on Internet and Multimedia Systems and Applications, Washington, USA, 2011.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In Proceedings of ACM CSCW'94 Conference on Computer Supported Cooperative Work, Sharing Information and Creating Meaning, pages 175–186, 1994.

[10] M. Pazzani and D. Billsus. Content-based Recommendation Systems In: The Adaptive Web, Vol.4321, pp. 325-341, 2007.

[11] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen. Collaborative Recommender Systems. In: The Adaptive Web, Vol.4321, pp.291-324, 2007.

[12] M. Deshpande, G. Karypis, Item-based Top-N Recommendation Algorithms, ACM Transactions on Information Systems. Vol.22, Issue 1, pp. 143 - 177, 2004.

[13] J.-Y. Hong, E.-h. Suh, and S.-j. Kim. Context-aware Systems: A Literature Review and Classification. Expert Systems with Applications, vol. 36, no. 4, May 2009.

[14] A. Chen. Context-Aware Collaborative Filtering System. Proc. of the International Workshop on Location- and Context-Awareness 2005, Springer LNCS 3479.

[15] G. Adomavicius et al. Incorporating contextual information in recommender system using a multidimensional approach. ACM Transactions on Information Systems, Vol. 23, No. 1, pp. 103-145J, 2005.

[16] M. Kirsch-Pinheiro, M. Villanova-Oliver, J. Gensel, and H. Martin. Context-Aware Filtering for Collaborative Web Systems: Adapting the Awareness Information to the Users Context. In: 20th ACM Symposium on Applied Computing (SAC 2005). ACM Press, Santa Fe, Nuevo Mexico, USA, 1668-1673, 2005.

[17] T. D. Pessemier, T. Deryckere, L .Martens. Context-Aware Recommendations for User-Generated Content on a Social Network Site. In Proceedings of the EuroITV '09 Conference, pp 133–136, New York, USA, 2009.

[18] Z. Yu, X. Zhou, D. Zhang, C-Y. Chin, X. Wang and J. Men. Supporting Context-Aware Media Recommendations for Smart Phones. IEEE Pervasive Computing, Vol.5, No.3, pp. 68-75, 2006.

[19] V. Setten, M. Pokraev and S. Koolwaaij. Context-Aware Recommendations on the Mobile Tourist Application. Nejdl, W. & De Bra, P. (Eds), LNCS 3137, Springer-Verlag, 2004.

[20] R. Hu, W. Dou and J. Liu. A Context-Aware Collaborative Filtering Approach for Service Recommendation. Proc. of 2012 International Conference on Cloud computing and Service Computing (CSC2012), IEEE Press, Nov. 2012.

[21] H. Si, Y. Kawahara, H. Kurasawa, H. Morikawa, and T. Aoyama. A context-aware collaborative filtering algorithm for real world oriented content delivery service. In Ubicom2005 Metapolis and Urban Life Workshop, 2005.

[22] K.K. Dhara and V. Krishnaswamy. Integrating multiple contexts in Real-time Collaborative Applications. Porc. Of the 3rd ACM Workshop on Context-Awareness in Retrieval and Recommendation (CaRR '13), ACM Press, pp7-14, Feb., 2013.

[23] G. Adomavicius, A. Tuzhilin. Towards the next Generation of Recommender Systems: A survey of the State-of-the-art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, 17 (6), 734-749, 2005.

[24] A. H. van Bunningen, L. Feng, and P. M. Apers. Context for ubiquitous data management. In Workshop on Ubiquitous Data Management, pages 17 – 24, Oct 2005.

[25] M. Kirsch-Pinheiro, M. Villanova-Oliver, J. Gensel, and H. Martin. Context-Aware Filtering for Collaborative Web Systems: Adapting the Awareness Information to the Users Context. In: 20th ACM Symposium on Applied Computing (SAC 2005). ACM Press, Santa Fe, Nuevo Mexico, USA, 1668-1673, 2005.

[26] T.D. Pessemier, S. Dooms L. Martens. Context-aware recommendations through context and activity recognition in a mobile environment. Multimed Tools Appl. doi: 10.1007/s11042-013-1582-x, 2013.

[27] S. Perugini and M. A. Gonçalves. Recommendation and personalization: a survey. Journal of Intelligent Information Systems, 2002.

[28] R. Burke. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, p.331-370, doi: 10.1023/A: 1021240730564, 2002.

[29] R. Burke, A. Felferning, and M. H. Goker, Recommender systems: an overview. AI Magazine, vol. 32, no. 3, pp.13–18, 2011.

[30] F. Ricci. Mobile recommender systems. Information Technology & Tourism, vol. 12, no. 3, pp. 205–231(27), 2010.

[31] Q. H. Mahmoud. Provisioning Context-aware Advertisements to Wireless Mobile Users. In Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME'06), pp. 669-672, 2006.

[32] T.G. Morrell, L. Kerschberg. Personal health explorer: A semantic health recommendation system. 28th International IEEE Conference on Data Eng. Workshops, pp. 55–59. doi:10.1109/ICDEW.2012.64, 2012.

[33] G. Tumas, F. Ricci. Personalized mobile city transport advisory system. In: ENTER Conference, 2009.

[34] J. H. Su, C., Kung, H. Yeh, P.S. Yu, V.S. Tseng. Music Recommendation Using Content and Context Information Mining. IEEE Intelligent Systems, vol. 25, no. 1, pp. 16 -26, 2010.

[35] J.K. Kim et al. A personalized recommender system for mobile commerce applications. J MIS Res 15(3):223–241, 2005.

[36] J. Ben Schafer, J. Konstan , J. Riedl. Recommender systems in e-commerce. Proceedings of the 1st ACM conference on Electronic commerce, p.158-166, November 03-05, 1999, Denver, Colorado, USA [doi>10.1145/336992.337035]

[37] Ko, S. -M. Choi, H. -S. Eom, J. -W. Cha, H. Cho, L. Kim, and Y. -S. Han. A smart movie recommendation system. Human Interface and the Management of Information. Interacting with Information, pp. 558–566, 2011.

[38] A.M Otebolaku, M. T. Andrade. Recognizing High-Level Contexts from Smartphone Built-in Sensors for Mobile Media Content Recommendation, in Proc of the 14th IEEE International Conference on Mobile Data Management, Milan Italy, 2013.

[39] T. Hussein, T. Linder, G. Werner G, and J. Ziegler. Hybreed: A software framework for developing context-aware hybrid recommender systems. User Model. User-Adapt. Interact, Special Issue on Context-aware Recommender Systems 24(1-2), 121–174, 2014.

[40] P. Umberto and G. Michele. Incorporating context into recommender systems: an empirical comparison of context-based approaches. Electronic Commerce Research, Volume.12, Issue.1, pp. 1, ISSN: 13895753, 2012.

[41] A. Costa, R. Guizzardi, J. Filho. COReS: Context-aware, Ontology-Based Recommender System for Service Recommendation. In Proceedings of the 19th international conference on advanced information systems engineering (CAiSE'07), Trondheim, Norway, pp 11-15, 2007.

[42] J. A Flanagan. Context-awareness in a mobile device: ontologies versus unsupervised/supervised learning. [on-line], Available from: http://www.cis.hut.fi/AKRR05/papers/akrr05flanagan.pdf

[43] C. Panatiotou and G. Samaras mPERSONA: Personalized portals for the wireless user: An agent approach Mobile Networks and Applications, 9 (6) pp. 663–677, 2004.

[44] D. Bouneffouf, A.Bouzeghoub, A. L. Gançarski. Following the User's Interests in Mobile Context-Aware Recommender Systems: The Hybrid-e-greedy Algorithm. WAINA, pp.657-662, 2012 26th International Conference on Advanced Information Networking and Applications Workshops, 2012.

[45] O. Kwon. I Know What You Need to Buy: Context-Aware Multimedia-based Recommendation System. Expert Systems with Applications. 25:387–400, 2003.

[46] J. Wang, J., Li, Z., Yao, J., Sun, Z., Li, M., Ma, W. Adaptive User Profile Model and Collaborative Filtering for Personalized News. In Proceedings of APWeb, pp.474-485, 2006.

[47] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Department of Computer Science, Dartmouth College, Technical Report TR2000-381, November 2000.

[48] A. B. Benitez, D. Zhong, S.F. Chang, J.R. Smit. MPEG-7 MDS Content Description Tools and Applications. W. Skarbek (Ed.): Computer Analysis of Images and Patterns, Lecture Notes in Computer Science 2124, Springer, 2001.

[49] G. J. F. Jones and P. J. Brown. Context-aware retrieval for ubiquitous computing environments. Mobile and Ubiquitous Information Access Workshop 2003, pages 227–243, 2004.

[50] M. Christos and P. Georgea. Ubiquitous Recommender Systems. Springer Journal of Computing, doi: 10.1007/s00607-013-0351-z, 2013.

[51] T. Ruotsalo et al. SMARTMUSEUM: A mobile recommender system for the Web of Data, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 20, Pp 50-67, 2013.

[52] W. Yin, X. Zhu, C. W. Chen. Contemporary ubiquitous media services: Content recommendation and adaptation. Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on, vol., no., pp.129, 134, 21-25 March 2011.

[53] W. Paireekreng and K. W. Wong. Adaptive Mobile Content Personalization Using Time-of-day. In 7th International Conference on e-Business (INCEB2008). Bangkok, Thailand, 2008.

[54] P. Resnick and H.R. Varian. Recommender Systems. Communications of the ACM 40(3): 56–58, 1997.

[55] R. V. Meteren and M. V. Someren. Using Content-Based Filtering for Recommendation. MLnet / ECML2000 Workshop, May 2000, Barcelona, Spain.

[56] S. Gauch , M. Speretta , A. Chandramouli and A. Micarelli. User profiles for personalized information access. The adaptive web: methods and strategies of web personalization, Springer-Verlag, Berlin, Heidelberg, 2007.

[57] G. Shani, L. Rokach, A. Meisles, L. Naamani, N. Piratla and D. Ben-Shimon. Establishing User Profiles in the MediaScout Recommender System. Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on, vol., no., pp.470- 476, March 1 2007-April 5 2007 doi: 10.1109/CIDM.2007.368912

[58] C. Yu, L. Lakshmanan, and S. Amer-Yahia. Recommendation diversification using explanations. In ICDE, 2009.

[59] L. Iaquinta, M. de Gemmis, P. Lops, G. Semeraro, M. Filannino, and P. Molino. Introducing Serendipity in a Content-based Recommender System, Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on 10-12, pp 168 – 173, Sept. 2008.

[60] J. L. Herlocker, J.A. Konstan, L.G. Terveen and J.T. Riedl. Evaluating Collaborative Filtering Recommender Systems. ACM Trans. Information Systems, vol. 22, no. 1, pp. 5-53, 2004

[61] B.M. Sarwar et al. Item-Based Collaborative Filtering Recommendation Algorithms. 10th Int',l World Wide Web Conference, ACM Press, pp. 285-295, 2001.

[62] Y. Koren , R.Bell , C. Volinsky. Matrix Factorization Techniques for Recommender Systems, Computer, v.42 n.8, p.30-37, August 2009 [doi>10.1109/MC.2009.263]

[63] X. Amatrian, A. Jaimes N. Oliver, J.M. Puriol. Data mining methods for recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor P (eds) Recommender systems handbook. Springer, Berlin, pp 39-71, 2011.

[64] H-J. Kwon; K-S. Hong. Personalized smart TV program recommender based on collaborative filtering and a novel similarity method. Consumer Electronics, IEEE Transactions on, vol.57, no.3, pp.1416-1423, August 2011. doi: 10.1109/TCE.2011.6018902.

[65] M. V. Setten. Supporting people in finding information. Hybrid recommender systems and goal-based structuring. Telematica Instituut Fundamental Research Series, vol. 016. Enschede, the Netherlands: Telematica Instituut, 2005 ISBN 90-75176-89-9

[66] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation.Communications of the ACM, 40(3):66-72, 1997.

[67] R. Burke. Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg, 2007.

[68] G. Mustansar and A. Prugel-Bennett. Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering. IAENG International Journal of Computer Science, 37(3), 2010.

[69] S. Bostandjiev, J. O'Donovan and T. Höllerer. TasteWeights: a visual interactive hybrid recommender system, Proceedings of the sixth ACM conference on Recommender systems, September 09-13, 2012, Dublin, Ireland [doi>10.1145/2365952.2365964]

[70] A. Ahmad and K. M. Ebrahimi. A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups Inform. Sci., 219 (2013), pp. 93–110.

[71] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete and M.A. Rueda-Morales. Combining content-based and collaborative recommendations: a hybrid approach based on Bayesian networks. International Journal of Approximate Reasoning, 51(7), pp.785–799, 2010.

[72] F. Aksel and A. Birtürk,. Enhancing Accuracy of Hybrid Recommender Systems through Adapting the Domain Trends. RecSys 2010 Workshop on the Practical use of Recommender Systems, Algorithms and Technologies (PRSAT2010), Sept. 2010, Barcelona, Spain.

[73] G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin. Context-Aware Recommender Systems. AI Magazine. 32(3), pp67-80, 2011.

[74] F. Meyer and F. Fessant. Reperio: A Generic and Flexible Industrial Recommender System, Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 502-505, 2011.

[75] A.B. Barragáns-Martínez, E. Costa-Montenegro, J.C. Burguillo, M. Rey-López, E.A. Mikic-Fonte, A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition Information Sciences, 180 (22) (2010), pp. 4290–4311

[76] E. García, C. Romero, S. Ventura, C.D. Castro. An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. User Modeling and User-Adapted Interaction 19 (1-2), 2009.

[77] F. Abel , R. Baumgartner , A. Brooks , C. Enzi , G. Gottlob , N. Henze , Îœ. Herzog , M. Kriesell , W. Nejdl and K. Tomaschewski. The Personal Publication Reader, Semantic Web Challenge. Proc. Fourth Int\'l Semantic Web Conf., 2005.

[78] K. Miyahara and M.J. Pazzani. Collaborative filtering with the simple Bayesian classifier. In Pacific Rim International Conference on Artificial Intelligence, 2000.

[79] M. H. Park, J. H. Hong and S. B. Cho. Location-based recommendation system using Bayesian user's preference model in mobile devices. Proceedings of the 4th international conference on Ubiquitous Intelligence and Computing, July 11-13, 2007, Hong Kong, China.

[80] T. D. Pessemier, T., Deryckere and L. Martens, L. Extending the Bayesian classifier to a context-aware recommender system for mobile devices. In Proceedings of fifth international conference on internet and web applications and services, Barcelona, Spain , pp. 242–247, 2010.

[81] M. Krstic and M. Bjelica. Context-aware personalized program guide based on neural network. Consumer Electronics, IEEE Transactions on, vol.58, no.4, pp.1301, 1306, doi: 10.1109/TCE.2012.6414999, 2012.

[82] C. Christakou, A. Stafylopatis. A hybrid movie recommender system based on neural networks. Intelligent Systems Design and Applications. ISDA '05. Proceedings. 5th International Conference on, vol., no., pp.500, 505, 8-10 Sept. 2005.

[83] A. Aamodt, E Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. In AI Communications 7, no. 1, 1994.

[84] I. Watson. Case-Based Reasoning is a Methodology not a Technology. Knowledge-based Systems 12: pp. 303-308, 1999.

[85] J. Lee and J. Lee. Context awareness by case-based reasoning in a music recommendation system. Ubiquitous Computing Systems, pages 45-58, 2008.

[86] O. Bouidghaghen, L. Tamine M. BoughaneM. Context-Aware User's Interests for Personalizing Mobile Search. 12th IEEE International Conference on Mobile Data Management (MDM), pp.129, 134, 6-9 June 2011.

[87] B.M. Sarwar, G. Karypis, J.A. Konstan, and J.Reidl. Application of dimensionality reduction in recommender system - a case study. In ACM WebKDD 2000 Web Mining for E-Commerce Workshop, 2000.

[88] R. Gupta, A. Jain, S. Rana, S. Singh. Contextual information based recommender system using Singular Value Decomposition. Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on, vol., no., pp.2084, 2089, 22-25 Aug. 2013.

[89] C. Hayes, P. Massa, P. Avesani and P. Cunningham. An online evaluation framework for recommender systems. In Workshop on Personalization and Recommendation in E-Commerce (Malaga, Spain, May 2002), Springer-Verlag.

[90] A. Said, D. Tikk, D and Y. Shi. Recommender Systems Evaluation: A 3D Benchmark. In: ACM RecSys 2012 Workshop on Recommendation Utility Evaluation: Beyond RMSE, Dublin, Ireland, pp. 21–23 (2012).

[91] J. A. Konstan and J. Riedl. Research resources for recommender systems. In CHI' 99 Workshop Interacting with Recommender Systems, 1999.

[92] H. Zhu et al. Mining Mobile User Preferences for Personalized Context-Aware Recommendation. ACM Transaction on Intelligent Systems and Technology, Vol, No, 2013.

[93] S. Anand and B. Mobasher. Contextual recommendation. In: Discovering and deploying user and content profiles, Springer Berlin/Heidelberg, pp 142–160, 2007.

[94] B. Schilit, N. Adams and R. Want. Context-aware computing applications. In: First International Workshop on Mobile Computing Systems and Applications, 85-90, 1994.

[95] A.K. Dey G.D. Abowd. Towards a better understanding of context and context-awareness. CHI'2000 Workshop on the What, Who, Where, When, and How of Context-Awareness, 2000, ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/ 99-22.pdf

[96] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni A survey of context modelling and reasoning techniques Pervasive and Mobile Computing, 6 (2), pp. 161–180, 2010.

[97] J. Kwapisz, G. Weiss, S. Moor. Activity Recognition Using Cell Phone Accelerometers. ACM SIGKDD Explorations Newsletter, 12 (2), 74-82, 2012.

[98] J .Ye, S. Dobson and S. McKeever. Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing, 8 (2), pp. 36–66, 2012.

[99] R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and predicting context by learning from user behavior. In Proceedings of the International Conference On Advances in Mobile Multimedia (MoMM2003). Austrian Computer Society (OCG), September 2003.

[100] O. D. Lara, A. J. Perez, M. A. Labrador, and J. D. Posada. Centinela: A human activity recognition system based on acceleration and vital sign data. Pervasive and Mobile Computing, vol.8 (2012), pp 717-729, 2012.

[101] P. Casale O. Pujol P. Radeva. Human Activity Recognition from Accelerometer Data using a Wearable Device. In Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), 289-296, 2012.

[102] N. Gyorbiro, A. Fabian & G. Homanyi. An Activity Recognition System for Mobile Phones. In Mobile Networks and Applications, 14 (1), 82-91, 2009.

[103] P. Siirtola and J. RöninG. Recognizing Human Activities User-independently on Smartphones Based on Accelerometer Data. International Journal of Interactive Multimedia and Artificial Intelligence, 1 (5): 38-45, 2012.

[104] D. Stefan, D. Barnan K. Narayanan, B.L. Thomas D.J. Cook. Simple and Complex Activity Recognition through Smart Phones, 2012 8th IEEE International Conference on Intelligent Environments (IE), pp. 214-221, 26-29, 2012.

[105] S.W. Lee and K. Mase. Activity and Location Recognition Using Wearable Sensors. IEEE Pervasive Computing, 1 (3): 24-32, 2002.

[106] U. Maurer, A. Smailagic, D. Siewiorek & M. Deisher. Activity Recognition and Monitoring using Multiple Sensors on different Body Positions. In IEEE Proceedings of the International Workshop on Wearable and Implantable Sensor Networks, 2006.

[107] F. Davide, C. Pedro, R.F. Diogo, & J. M. Cardoso. Preprocessing Techniques for Context Recognition from Accelerometer Data. Personal and Ubiquitous Computing, 14(7), 645-662, 2010.

[108] R.T. Olszewski, C. Faloutsos, D.B. Dot. Generalized feature extraction for structural pattern recognition in time-series data. Tech. rep., In Time-SeriesData, Ph.D. dissertation, Carnegie Mellon University, 2001.

[109] M. Marcu N. Ghiata, V. Cretu. Extracting high-level user context from low-level mobile sensors data. Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium, vol., no., pp.449, 454, 23-25 May, 2013.

[110] O.D.Lara, M.A. Labrador. A Survey on Human Activity Recognition using Wearable Sensors. Communications Surveys & Tutorials. IEEE, vol.15, no.3, pp.1192, 1209, Third Quarter 2013,doi: 10.1109/SURV.2012.110112.00192, 2013.

[111] T. Mashita, D. Komak M. Iwata, K. Shimatani, H. Miyamoto T. Hara et al. A Content Search System for Mobile Devices Based on User Context Recognition. In Proceedings of the International Conference on Virtual Reality, pp 1-4, 2012.

[112] G. Milette and A. Stroud. Professional Android Sensor Programming, 1st Edition, John Willey and Sons Inc, 2012.

[113] Y. Kwon, K. Kang and C. Bae. Unsupervised learning for human activity recognition using smartphone sensors, Expert Systems with Applications, Volume 41, Issue 14, 15 October 2014, Pages 6067-6074, ISSN 0957-4174, http://dx.doi.org/10.1016/j.eswa.2014.04.037.

[114] L. Bao and S.S. Intille. Activity Recognition from User Annotated Accelerometer Data. Pervasive Computing, LNCS Vol. 3001, 1-17, 2004.

[115] Consolvo S., David W., McDonald D. W., Toscos T. , Chen M.Y., Froehlich J. et al. Activity Sensing in the Wild: A Field Trial of UBiFiT Garden. Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, April 05-10, 2008, pp. 1797, Florence, Italy.

[116] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, et al. Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, November 05-07, 2008, Raliegh, USA.

[117] J. W Lockhart, T. Pulickal and G. M.Weiss. Applications of Mobile Activity Recognition in proceedings of the ACM UbiComp International Workshop on Situation, Activity, and Goal Awareness, Pittsburgh, PA, 2012.

[118] S. L. Lau and K. David K. Movement Recognition using the Accelerometer in Smartphones, Future Network and Mobile Summit, pp. 1-9, 2010.

[119] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey On Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4), pp.263-277, June 2007.

[120] A. Gómez-Pérez, M. Fernández-López and O. Chorcho. Ontological Engineering. Springer-Verlag, London, UK, 2004.

[121] J. Indulska, R. Robinsona., A. Rakotonirainy and K Henricksen. Experiences in using cc/pp in context-aware systems. In LNCS 2574: Proceedings of the 4th International Conference on Mobile Data Management (MDM2003)(Melbourne/Australia. M.-S. Chen, P. K. Chrysanthis, M. Sloman, and A. Zaslavsky, Eds., Lecture Notes in Computer Science (LNCS), Springer,pp. 247–261, 2003.

[122] K.F.Yeung, Y. Yang and D. Ndzi. A proactive, personalized mobile recommendation system using the analytic hierarchy process and Bayesian network. J. Internet Services and Applications, 195–214, 2012.

[123] OMA Open Mobile Alliance (OMA), “User Agent Profile (UAProf)”. OMA-TS-UAProf-V2 0-20060206-A, 2006.

[124] B. L. Tseng, C. Lin and J. R. Smith. Using MPEG-7 and MPEG-21 for Personalizing Video. IEEE Multimedia, Vol.11, Issue 1, pp. 42-52, Jan-March, 2004.

[125] O. Steiger, T. Ebrahmi and D. Marimon. MPEG-based Personalized Content Delivery. Proceedings of the 2003 International Conference on Image Processing (ICIP), Barcelona, Spain, September 2003.

[126] M. Angelides, A. Sofokleous and C. Schizas. Mobile Computing with MPEG-21. Proceedings of the International Symposium on Ubiquitous Intelligence and Smart Worlds (UISW2005), pp 556-565, Nagasaki, Japan, December 2005.

[127] M.T. Andrade, S. Dogan, A. Carreras, V. Barbosa, H.K. Arachchi, J. Delgado, A. M.Kondoz. Advanced Delivery of Sensitive Multimedia Content for better serving User Expectations in Virtual Collaboration Applications. Multimedia Tools Applications. 58(3): pp. 633-661, 2012.

[128] W. Y. Lum and F. C.M. Lau. A Context-Aware Decision Engine for Content Adaptation. IEEE Pervasive Computing, vol. 1, no. 3, pp. 41-49, July-Sept. 2002.

[129] C-H. Liu, K.L Chang, J.Y. Jason and S.C. Hung. Ontology-Based Context Representation and Reasoning Using OWL and SWRL. 8th Annual Communication Networks and Services Research Conference, Montreal, Quebec, Canada, 2010.

[130] R.A. Carrillo., M. Villanova-Oliver, J. Gensel and H. Martin. Contextual User Profile for adapting information in nomadic environments. In: Proc. of the 1st Workshop on Personalized Access to Web Information (PAWI 2007), Nancy, France, December 3rd, 2007. LNCS, Springer, Heidelberg.

[131] G. Adomavicius and A. Tuzhilin. Expert-Driven Validation of Rule-Based User Models in Personalization Applications. Data Mining and Knowledge Discovery, vol. 5, nos. 1 and 2, pp. 33-58, 2001.

[132] D. Vallet, M. Ferná,ndez and P. Castells. An Ontology-Based Information Retrieval Model. Proc. Second European Semantic Web Conf. (ESWC ',05), 2005.

[133] M. Sutterer, O. Droegehorn and K. David. User profile selection by means of ontology reasoning. In: Proceedings of the 2008 fourth advanced international conference on telecommunications, pp 299–304, 2008.

[134] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In Workshop on Context-Aware Recommender Systems (CARS 2009). New York, 2009.

[135] L. Baltrunas and F. Ricci. Context-dependent items generation in collaborative filtering. In RecSys'09 workshop 3: CARS, 2009.

[136] K. Yu, B. Zhang, H. Zhu, H. Cao, and J. Tian. Towards personalized context-aware recommendation by mining context logs through topic models. In PAKDD'12, 2012.

[137] C.V. Ostuni et al. Mobile Movie Recommendation with Linked Data. LNCS Vol.8127, pp.400-415, 2013.

[138] K. Oku, S. Nakajima, J. Miyazaki, and Uemura, S. Context-aware SVM for the context dependent information recommendation. In Proceedings of the 7th International Conference on Mobile Data Management, page 109, 2006.

[139] C. Timmerer, T. DeMartini, and H. Hellwagner. The MPEG-21 Multimedia Framework: Conversions and Permissions. Proc. DACH Security 2006, pp. 225-235, Düsseldorf, Germany, Mar. 2006.

[140] Y. Wang, J. G. Kim, Shih-Fu Chang, Hyung-Myung Kim. Utility-Based Video Adaptation for Universal Multimedia Access (UMA) and Content-Based Utility Function Prediction for Real-Time Video Transcoding. IEEE Transactions on Multimedia, Vol. 9, No. 2, February 2007.

[141] A. Doulamis and G. Tziritas. Content-based Video Adaptation in Low/Variable Bandwidth Communication Networks Using Adaptable Neural Network Structures. 2006 International Joint Conference on Neural Networks, Sheraton Vancouver.

[142] D. Jannach, K. Leopold, C. Timmerer, and H. Hellwagner. A Knowledge-based Framework for Multimedia Adaptation. Applied Intelligence, vol. 24, no. 2, pp. 109- 125, April 2006.

[143] B. Shao, M. Mattavelli, D. Renzi, M.T. Andrade, S. Battista, S. Keller, G. Ciobanu, P. Carvalho. A multimedia terminal for adaptation and End-to-End QoS control. Multimedia and Expo, 2008 IEEE International Conference, June 23 2008- Pp433 – 436, April 26 2008.

[144] M. Mackay, M. Ransburg, D. Hutchison, and H. Hellwagner. Combined Adaptation and Caching of MPEG-4 SVC in Streaming Scenarios. Proc. of the 9th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2008), Klagenfurt, Austria, May 2008.

[145] M. Prangl, T. Szkaliczki, and H. Hellwagner: A Framework for Utility-based Multimedia Adaptation. IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 6, pp. 719-728, Jun. 2007.

[146] M. T. Andrade et al. Using Context to Assist the Adaptation of Protected Multimedia Content in Virtual Collaboration Applications. IEEE/ICST 3rd International Conference on Collaborative Computing (CollaborateCom 2007), New York, November 2007.

[147] C. Timmerer. MPEG-21 Digital Item Adaptation. FDAM/2, Dynamic and Distributed Adaptation, 78th ISO/IEC JTC 1/SC 29/WG 11 (MPEG) Meeting, Hangzhou, China, 2006.

[148] A.A. Sofokleous, and M.C. Angelides. DCAF: An MPEG-21 Dynamic Content Adaptation Framework. Multimedia Tools and Applications 40 (2), 151–182, 2008.

[149] Shih-Fu Chang Anthony Vetro. Video Adaptation: Concepts, Technologies and Open Issues. Technical Report Mitsubishi Electric Research Laboratories, TR-2005-010 January 2005.

[150] M. Huebscher and J. McCann A Survey of Autonomic Computing- degrees, models and applications *ACM Transactions Autonomous Adaptive Systems (TAAS)*, December 2006.

[151] S. Dobson, S. Denazis, A. Fernandez, D. GaÄ³ti, E. Gelenbe, F. Massacci, P. Nixon, F. Sa®re, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. ACM Transactions Autonomous Adaptive Systems (TAAS), 1(2):223{259, December 2006.

[152] S-F. Chang, T. Sikora and A. Puri. Overview of the MPEG-7 standard. IEEE Trans. on Circuits and Systems for Video Technology 11(6), 688–695, 2001.

[153] V. Barbosa and M.T. Andrade. MULTICAO: A semantic approach to context-aware adaptation decision taking. Image Analysis for Multimedia Interactive Services, 2009. WIAMIS '09. 10th Workshop on, vol., no., pp.133-136, 6-8 May 2009.

[154] T. Lemlouma and N. Layaïda. The Negotiation of Multimedia Content Services in Heterogeneous Environments. In the MMM 2001: the 8th International Conference on Multimedia Modeling, CWI, Amsterdam, Netherlands, 5-7 November 2001. pp. 187-206.

[155] M. P. Ruiz, J. A. Botía, A. Gómez-Skarmeta. Providing QoS through machine-learning-driven adaptive multimedia applications. IEEE Transactions on Systems, Man,and Cybernetics—Part B: Cybernetics, Vol. 34, No. 3, June 2004.

[156] O. Coutand, S. Haseloff, S.L., Lau, K. David. A Case-based Reasoning Approach for Personalizing Location-aware Services. In: Workshop on Case-based Reasoning and Context Awareness, 2006.

[157] C. Rack, S. Arbanowski and S. Steglich. A Generic Multipurpose recommender System for Contextual Recommendations. Autonomous Decentralized Systems, 2007. ISADS '07. Eighth International Symposium on, vol., no., pp.445- 450, 21-23 March 2007 doi: 10.1109/ISADS.2007.2

[158] V. Bellotti, B. Begole, E.H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M.W. Newman, K. Partridge, B. Price, P. Rasmussen, M. Roberts, D.J. Schiano, A. Walendowski. Activity-based serendipitous recommendations with the Magitti mobile leisure guide. Proc. of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems, CHI '08, ACM, New York, NY, USA (2008), pp. 1157–1166.

[159] D.L. Nicholas, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A. T. Campbell. A survey of mobile phone sensing, IEEE Communications Magazine, v.48 n.9, p.140-150, September 2010 [doi>10.1109/MCOM.2010.5560598]

[160] I. Burnett, F. Perreira, R. Vande Walle and R.Koene. The MPEG-21 Book. John Wiley and Sons Ltd,, 2006.

[161] A. Salden et al. Contextual Personalization of a Mobile Multimodal Application. In Proceedings of the International Conference on Internet Computing, ICOMP 2005, Las Vegas, Nevada, USA, June 2005.

[162] M. Perttunen, J. Riekki and O. Lassila. Context representation and reasoning in pervasive computing: a review. International Journal of Multimedia and Ubiquitous Engineering 4(4), pp1–28, 2009.

[163] W3C, Web Ontology Language (OWL). Available at http://www.w3.org/TR/owl-features/

[164] S. Ait Chellouche, D. Négru. Context-aware multimedia service provisioning in future Internet using ontology and rules. 8th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, 2011.

[165] SWRL: A Semantic Web Rule Language Combining OWL and RuleMLhttp://www.w3.org/Submission/SWRL/

[166] V. Balakrishnan, M. R. Shakouri, H. Hoodeh, L. Huck-Soo. Classification Algorithms in Human Activity Recognition Using Smartphones. International Journal of Computer and Information Engineering, pp77-84, Vol. 6, 2012.

[167] A.M. Khan, Y-K. Lee, S.Y.Lee, and T-S. Kim. A Triaxial Accelerometer-Based Physical-Activity Recognition via Augmented-Signal Features and a Hierarchical Recognizer. Information Technology in Biomedicine, IEEE Transactions on, vol. 14, no. 5, pp. 1166-1172, Sept, 2010.

[168] M. Abdullah, A. Negara, S. Sayeed and K. Muthu. Classification Algorithms in Human Activity Recognition Using Smartphones, International journal of Computer and Information Engineering, Issue 6, pp. 422-429, 2012.

[169] N. Ravi, N. Dandekar, P. Mysore and L. Littman. Activity Recognition from Accelerometer Data, Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence, 2005.

[170] A. Mannini and A.M. Sabatini. Machine Learning Methods for Classifying Human Physical Activity from Onbody Accelerometers. In Sensors, 10, 1154—1175, 2010.

[171] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, 1993.

[172] I. H. Witten, E. Frank. Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. Morgan Kaufmann, 2005.

[173] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten. The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 2009.

[174] C. Timmerer., A. Vetro and H. Hellwagnera. Mpeg-21 Digital Item Adaptation. In Encyclopedia of Multimedia 2nd ed., 2008

[175] X. H. Wang, D. Q. Zhang T. Gu and H.K. Pung. Ontology based context modeling and reasoning using OWL. Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, vol., no., pp.18-22, 14-17 March 2004 doi: 10.1109/PERCOMW.2004.1276898.

[176] L. Liu, N. Mehandjiev and L. Xu. Using Context Similarity for Service Recommendation. In Proceedings of the fourth IEEE international conference on semantic computing, 2010.

[177] R. Krummenacher and T. Strang. Ontology-Based Context-Modeling. In Third Workshop on Context Awareness for Proactive Systems (CAPS'07), 2007.

[178] Time Ontology:http://www.w3.org/TR/owl-time/

[179] A.M. Otebolaku, M. T. Andrade. Context-Aware User Profiling and Multimedia Content Classification for Smart Devices. In the proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014). May 13-16, 2014, Victoria, Canada.

[180] A. Mitseva et al. Towards adaptive security for convergent wireless sensor networks in beyond 3G environments. Wireless Communications and Mobile Computing Journal, Vol.10 (9) Pp 1193-1207. John Wiley & Sons, Ltd.DOI: 10.1002/wcm.678, 2010.

[181] P. Melville, R.J. Mooney and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. Proc. 18th Nat',l Conf. Artificial Intelligence, 2002.

[182] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, Recommender Systems Handbook, pages 73-105. Springer Verlag, 2011.

[183] R. Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.

[184] R. Pressman. Software Engineering: A Practitioner's Approach, 5th ed. McGraw-Hill, USA, 2001. ISBN 0-07-365578-3.

[185] Oracle Java EE Technologies: http://www.oracle.com/technetwork/java/javaee/tech/

[186] Google Android Tutorial http://java.dzone.com/articles/google-android- tutorial

[187] M. Gao and W. Zhongfu. Personalized context-aware collaborative filtering based on neural network and slope one. Lecture Notes in Computer Science, vol. 5738, pp. 109-116, 2009.

[188] J. Yang. Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phone. The 1st International Workshop on Interactive Multimedia for Consumer Electronics (IMCE), ACM Multimedia Devices, pp.1-5, 25-29,2009 .

[189] Android Sensors: http://developer.android.com/guide/topics/sensors/sensors_motion.html.

[190] T. Huynh and B. Schiele. Analyzing Features for Activity Recognition. Proceedings of the Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies, October 12-14, Grenoble, France, 2005.

[191] J.W. Jamie, P. Lukowicz, H. Gellersen, Performance Metrics for Activity Recognition. ACM Transactions on Intelligent Systems and Technology (TIST), v. 2 n. 1, p. 1-23 2011.

[192] K. Stefanidis, E. Pitoura, and P. Vassiliadis. Adding Context to Preferences. In Proceedings of the International Conference on Data Engineering, ICDE, 2007.

[193] Z. Prekopcsák, S. Soha T. Henk and C. Gáspár-Papanek. Activity Recognition for Personal Time Management. Proceedings of the European Conference on Ambient Intelligence, pp. 55-59, 2009.

[194] J .L. Herlocker, J. A. Konstan, L.G. Terveen and J.T. Riedl. Evaluating Collaborative Filtering Recommender Systems. ACM Trans. Information Systems, vol. 22, no. 1, pp. 5-53, 2004.

[195] G. Shani and A Gunawardan. Evaluating recommendation systems. In: Ricci F, Rokach L, Shapira B, Kantor P (eds) Recommender systems handbook. Springer, Berlin, pp 257-298, 2011.

[196] X. Wang, D. Rosenblum and Y. Wang. Context-aware mobile music recommendation for daily activities. Proceedings of the 20th ACM international conference on Multimedia, October 29-November 02, 2012, Nara, Japan.

[197] K. Oku, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and F. Hattori. A Recommendation System Considering Users' Past/Current/Future Contexts. ids.csom.umn.edu, pp. 3-7, 2010. [Online].

[198] Y. Koren. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In ACM SIGKDD, pages 426–434, 2008.

[199] A.M. Otebolaku, M. T. Andrade. A context-aware framework for media recommendations on smartphones. In proceedings of the European Conference on the Use of Modern Information and Communication Technologies(ECUMICT 2014), Gent, March 2014. 27th-28th of March 2014, Gent, Belgium, Published by Springer Lecture Notes in Electrical Engineering, edited by De Strycker, Lieven,Vol. 302, pp 87-108.

[200] A.M. Otebolaku and M.T. Andrade. Context-Aware Media Recommendations for Smart Devices. Springer Journal of Ambient Intelligence and Humanized Computing, 2014, accepted

[201] A.M. Otebolaku and M.T. Andrade. Towards Designing a Software Platform for Context-Aware Mobile Content Personalization. (Under review process)

[202] A. M. Otebolaku and M. T. Andrade. Context-Aware Media Recommendations. In the proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014), May 13-16, 2014, Victoria, Canada.

[203] P. TalebiFard and V.C. Leung. A dynamic context-aware access network selection for handover in heterogeneous network environments. In Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, pp. 385-390, 2011.

[204] Z. Prekopcsák, S. Soha T. Henk and C. Gáspár-Papanek. Activity Recognition for Personal Time Management. Proceedings of the European Conference on Ambient Intelligence, pp. 55-59, 2009.

[205] A.G. Wilde. An overview of human activity detection technologies for pervasive systems, Department of Informatics University of Fribourg, Switzerland, 2010.

Appendix A

CAMR Client Evaluation Screenshots

Figure A.1 CAMR on Smartphone

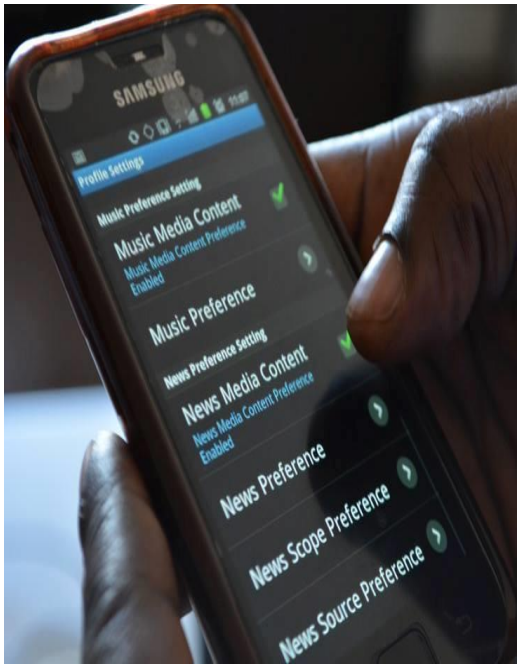


Figure A.2 CAMR during a demo

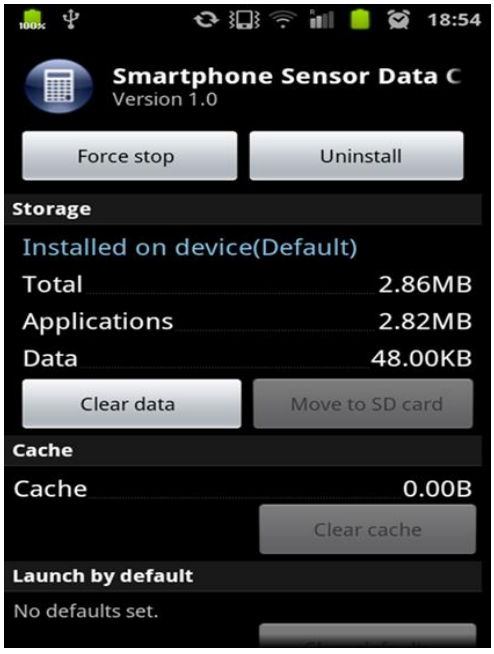


Figure A3. CAMR Context Collector Size on the Device

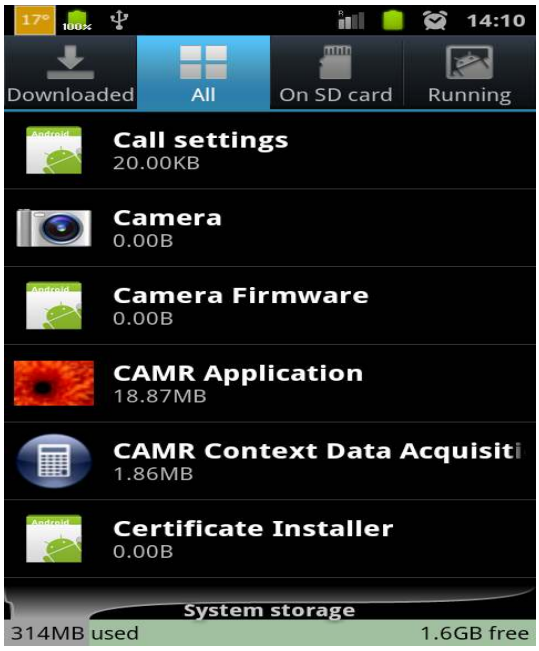


Figure A.4 CAMR and Context Service on the Device Default Screen

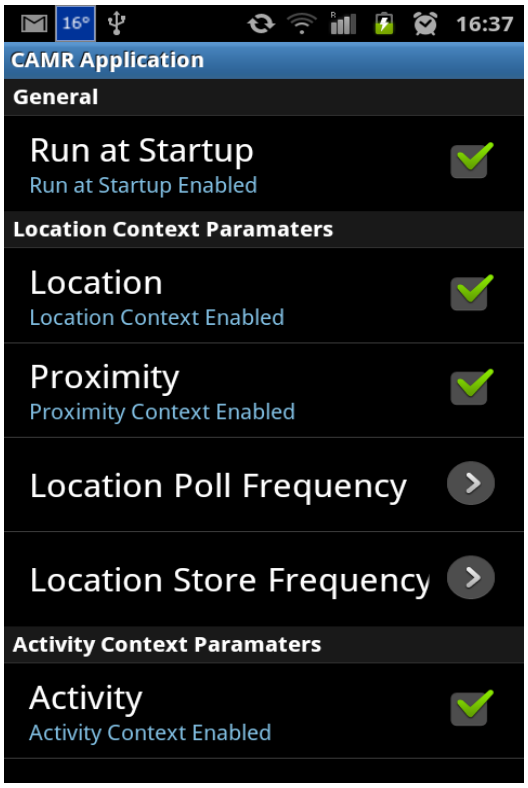


Figure A.5 CAMR Context Poll Setting

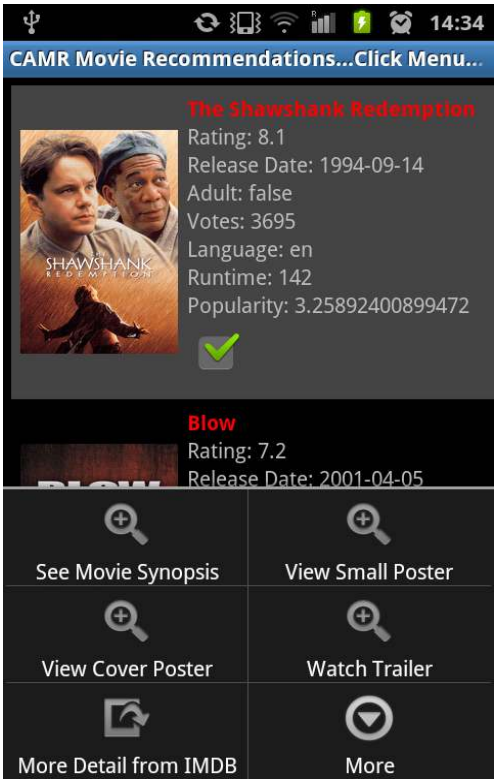


Figure A.6 Recommendation Set

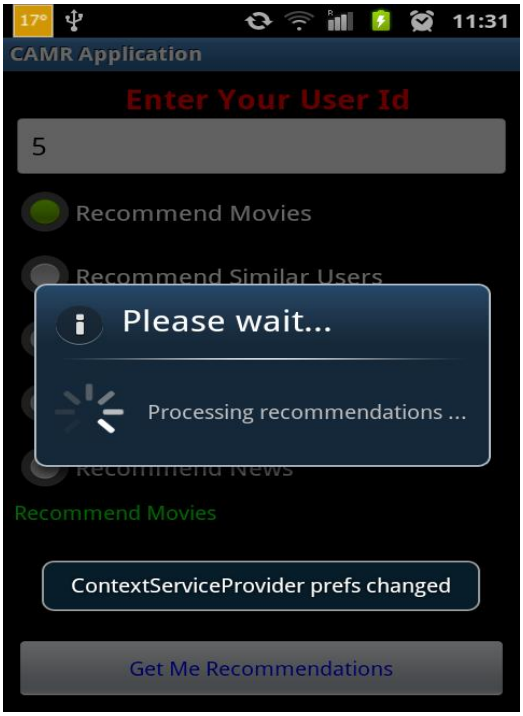


Figure A.7 Explicit Recommendations

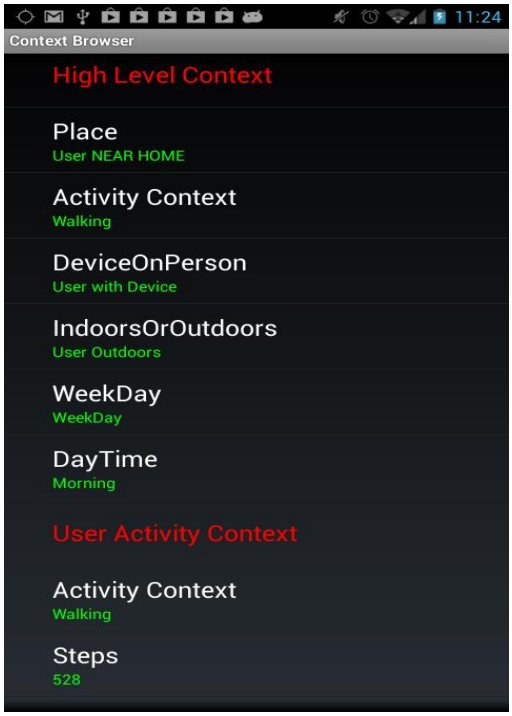


Figure A.8 More context information

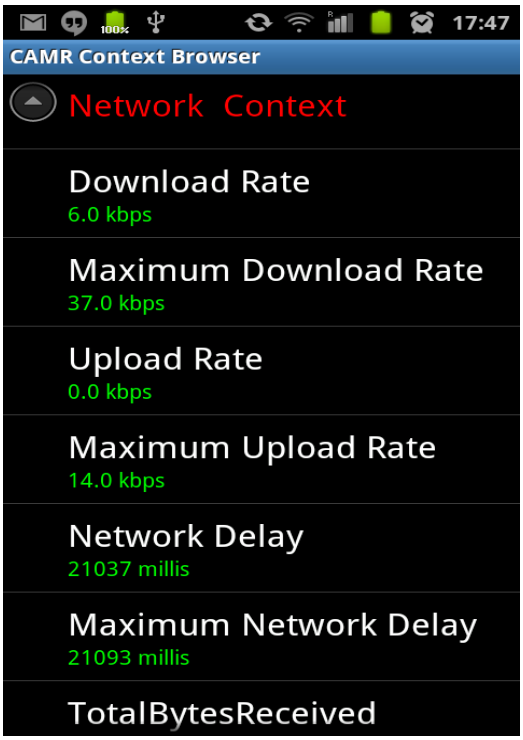


Figure A. 9 Network Conditions

Figure A.10 Weather conditions

Appendix B

Context Recognition Accuracies of Simple and Complex Activities of Various Classification Model

Table B.1: Multilayer Perceptron Confusion Matrix

Predicted Class								**Actual Class**
lying	**Driving**	**Running**	**Jogging**	**Walking**	**Sitting**	**Standing**		
1373	1	1	1	0	20	1	lying	
3	**310**	0	0	3	1	0	Driving	
0	0	**246**	7	3	0	0	Running	
0	0	2	**465**	4	0	0	Jogging	
6	5	10	4	**2177**	11	0	Walking	
25	6	0	1	14	**2119**	1	Sitting	
2	0	0	0	2	1	**259**	Standing	

Table B. 2:Random Forest Confusion Matrix

Predicted Class								**Actual Class**
lying	**Driving**	**Running**	**Jogging**	**Walking**	**Sitting**	**Standing**		
1386	1	2	0	1	7	0	lying	
0	**314**	0	0	3	0	0	Driving	
0	0	**247**	6	3	0	0	Running	
0	0	4	**461**	6	0	0	Jogging	
6	7	0	6	**2190**	4	0	Walking	
19	2	0	1	14	**2130**	0	Sitting	
2	0	0	0	2	1	**259**	Standing	

Table B. 3:PART Confusion Matrix

Predicted Class								Actual Class
lying	**Driving**	**Running**	**Jogging**	**Walking**	**Sitting**	**Standing**		
1363	4	0	0	7	21	2	lying	
3	**305**	0	0	5	3	1	Driving	
1	0	**245**	10	0	0	0	Running	
0	0	7	**450**	12	1	1	Jogging	
8	8	1	10	**2165**	18	3	Walking	
19	2	0	1	14	**2130**	0	Sitting	
2	0	0	0	2	1	**259**	Standing	

Table B. 4:C4.5 Confusion Matrix

Predicted Class								
lying	**Driving**	**Running**	**Jogging**	**Walking**	**Sitting**	**Standing**		**Actual Class**
1368	1	0	0	2	25	1	lying	
2	**305**	0	0	4	4	2	Driving	
0	0	**239**	15	1	1	0	Running	
0	0	12	**445**	14	0	0	Jogging	
1	8	0	12	**2183**	7	2	Walking	
19	2	0	1	14	**2130**	0	Sitting	
2	0	0	0	2	1	**259**	Standing	

Table B. 5: Confusion Matrix for Location and Activity Predictions using MLP

Predicted Class											
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***		**Actual Class**
2129	9	3	8	3	4	5	25	6	4	*A=Sitting/Home/Office*	
4	**200**	13	4	2	2	9	26	4	8	*B=Standing/Bus*	
8	8	**210**	18	0	4	6	47	2	18	*C=Standing/Train*	
1	2	32	**142**	8	2	1	20	2	2	*D=Ascending/Elevator*	
2	3	3	16	**40**	0	0	12	0	5	*E=Descending/Elevator*	
3	1	6	0	0	**108**	0	10	9	0	*F=DescendingStairs/Home/Office*	
7	13	4	3	1	1	**150**	16	3	3	*G=Sitting/Bus*	
23	19	35	11	7	5	9	**616**	10	9	*H=Standing/Home/Office*	
5	6	0	1	0	12	8	9	**126**	4	*I=AscendingStairs/Home/Office*	
7	12	33	8	4	2	1	30	9	**193**	*J=Sitting/Train*	

Table B. 6: Confusion Matrix for Location and Activity Predictions using Random Forest

Predicted Class											
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***		**Actual Class**
2177	4	1	2	0	0	3	5	3	1	*A=Sitting/Home/Office*	
5	**232**	5	0	1	1	2	12	4	6	*B=Standing/Bus*	
6	11	**258**	3	1	1	2	23	2	14	*C=Standing/Train*	
0	0	17	**173**	2	0	3	10	0	7	*D=Ascending/Elevator*	
0	1	5	13	**56**	0	0	2	0	4	*E=Descending/Elevator*	
2	3	4	0	0	**112**	2	3	11	0	*F=DescendingStairs/Home/Office*	
5	17	2	1	0	1	**165**	5	3	2	*G=Sitting/Bus*	
6	13	14	6	1	2	9	**685**	7	1	*H=Standing/Home/Office*	
3	5	0	0	0	8	5	2	**146**	2	*I=AscendingStairs/Home/Office*	
7	12	33	8	4	2	1	30	9	**193**	*J=Sitting/Train*	

Table B. 7: Confusion Matrix for Location and Activity Predictions using PART

Predicted Class											
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***		**Actual Class**
2138	8	6	3	1	3	9	18	4	6	*A=Sitting/Home/Office*	
12	**187**	14	2	1	4	12	20	8	12	*B=Standing/Bus*	
4	10	**218**	8	3	5	7	42	2	22	*C=Standing/Train*	
4	1	13	**148**	10	0	4	21	1	10	*D=Ascending/Elevator*	
2	0	3	9	**54**	1	2	2	0	8	*E=Descending/Elevator*	
2	3	4	0	1	**102**	2	7	16	0	*F=DescendingStairs/Home/Office*	
9	12	9	3	2	2	**143**	15	5	1	*G=Sitting/Bus*	
20	22	31	16	5	2	10	**614**	8	16	*H=Standing/Home/Office*	
3	5	0	0	0	8	5	2	**146**	2	*I=AscendingStairs/Home/Office*	
7	12	33	8	4	2	1	30	9	**193**	*J=Sitting/Train*	

Table B. 8: Confusion Matrix for Location and Activity Predictions using C4.5

Predicted Class											
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***		**Actual Class**
2139	7	7	4	1	5	12	15	3	3	*A=Sitting/Home/Office*	
6	**195**	10	4	1	1	21	20	4	10	*B=Standing/Bus*	
5	10	**223**	16	1	6	5	34	0	21	*C=Standing/Train*	
2	3	19	**143**	11	1	5	19	0	9	*D=Ascending/Elevator*	
5	4	2	20	**41**	0	1	3	0	5	*E=Descending/Elevator*	
4	2	7	0	0	**104**	0	5	15	0	*F=DescendingStairs/Home/Office*	
1	23	3	4	0	3	**136**	13	4	4	*G=Sitting/Bus*	
2	19	37	13	0	5	18	**618**	6	16	*H=Standing/Home/Office*	
3	5	0	0	0	8	5	2	**146**	2	*I=AscendingStairs/Home/Office*	
7	12	33	8	4	2	1	30	9	**193**	*J=Sitting/Train*	

Table B. 9: Confusion Matrix for Location and Activity Predictions using Naïve Bayes

Predicted Class											
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***		**Actual Class**
2139	7	7	4	1	5	12	15	3	3	*A=Sitting/Home/Office*	
6	**195**	10	4	1	1	21	20	4	10	*B=Standing/Bus*	
5	10	**223**	16	1	6	5	34	0	21	*C=Standing/Train*	
2	3	19	**143**	11	1	5	19	0	9	*D=Ascending/Elevator*	
5	4	2	20	**41**	0	1	3	0	5	*E=Descending/Elevator*	
4	2	7	0	0	**104**	0	5	15	0	*F=DescendingStairs/Home/Office*	
1	23	3	4	0	3	**136**	13	4	4	*G=Sitting/Bus*	
2	19	37	13	0	5	18	**618**	6	16	*H=Standing/Home/Office*	
3	5	0	0	0	8	5	2	**146**	2	*I=AscendingStairs/Home/Office*	
7	12	33	8	4	2	1	30	9	**193**	*J=Sitting/Train*	