# Web-Page Recommendation Based on Web Usage and Domain Knowledge

Thi Thanh Sang Nguyen, Hai Yan Lu, and Jie Lu

**Abstract**—Web-page recommendation plays an important role in intelligent Web systems. Useful knowledge discovery from Web usage data and satisfactory knowledge representation for effective Web-page recommendations are crucial and challenging. This paper proposes a novel method to efficiently provide better Web-page recommendation through semantic-enhancement by integrating the domain and Web usage knowledge of a website. Two new models are proposed to represent the domain knowledge. The first model uses an ontology to represent the domain knowledge. The second model uses one automatically generated semantic network to represent domain terms, Web-pages, and the relations between them. Another new model, the conceptual prediction model, is proposed to automatically generate a semantic network of the semantic Web usage knowledge, which is the integration of domain knowledge and Web usage knowledge. A number of effective queries have been developed to query about these knowledge bases. Based on these queries, a set of recommendation strategies have been proposed to generate Web-page candidates. The recommendation results have been compared with the results obtained from an advanced existing Web Usage Mining (WUM) method. The experimental results demonstrate that the proposed method produces significantly higher performance than the WUM method.

**Index Terms**—Web usage mining, Web-page recommendation, domain ontology, semantic network, knowledge representation

## 1 Introduction

Web-page recommendation has become increasingly popular, and is shown as links to related stories, related books, or most viewed pages at websites. When a user browses a website, a sequence of visited Web-pages during a session (the period from starting, to existing the browser by the user) can be generated. This sequence is organized into a Web session $S = d_1 d_2 \dots d_k$, where $d_i$ $(i = [1 \dots k])$ is the page ID of the $i$th visited Web-page by the user. The objective of a Web-page recommender system is to effectively predict the Web-page or pages that will be visited from a given Web-page of a website.

There are a number of issues in developing an effective Web-page recommender system, such as how to effectively learn from available historical data and discover useful knowledge of the domain and Web-page navigation patterns, how to model and use the discovered knowledge, and how to make effective Web-page recommendations based on the discovered knowledge.

A great deal of research has been devoted to resolve these issues over the past decade. It has been reported that the approaches based on tree structures and probabilistic models can efficiently represent Web access sequences (WAS) in the Web usage data [1]. These approaches learn from the training datasets to build the transition links between Web-pages. By using these approaches, given the current visited Web-page (referred to as a state) and $k$ previously visited pages (the previous $k$ states), the Web-page(s) that will be visited in the next navigation step can be predicted. The performance of these approaches depends on the sizes of training datasets. The bigger the training dataset size is, the higher the prediction accuracy is. However, these approaches make Web-page recommendations solely based on the Web access sequences learnt from the Web usage data. Therefore, the predicted pages are limited within the discovered Web access sequences, i.e., *if a user is visiting a Web-page that is not in the discovered Web access sequence, then these approaches cannot offer any recommendations to this user*. We refer to this problem as "new-page problem" in this study.

Some studies have shown that semantic-enhanced approaches are effective to overcome the new-page problem [2], [3] and have therefore become far more popular. The use of domain knowledge can provide tremendous advantages in Web-page recommender systems [4]. A domain ontology is commonly used to represent the semantics of Web-pages of a website. It has been shown that integrating domain knowledge with Web usage knowledge enhances the performance of recommender systems using ontology-based Web mining techniques [4]–[6]. Integrating semantic information with Web usage mining achieved higher performance than classic Web usage mining algorithms

- *T. T. S. Nguyen and H. Y. Lu are with the Decision Systems & e-Service Intelligence (DeSI) Lab, Centre for Quantum Computation & Intelligent Systems (QCIS), School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW 2007, Australia.*
  *E-mail: thi.t.nguyen-9@student.uts.edu.au, haiyan.lu@uts.edu.au.*
- *J. Lu is the Head of the School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW 2007, Australia. E-mail: jie.lu@uts.edu.au.*

*Manuscript received 5 July 2012; revised 14 Apr. 2013; accepted 26 Apr. 2013. Date of publication xxx. Date of current version xxx.*
*Recommended for acceptance by B. Cooper.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier 10.1109/TKDE.2013.78*

[5]–[9]. However, one of the big challenges that these approaches are facing is the semantic domain knowledge acquisition and representation. How to effectively construct the domain ontology is an ongoing research topic.

The domain ontology can be constructed manually by experts, or by automatically learning models, such as the Bayesian network or a collocation map, for many different applications. Manually building an ontology of a website is challenging given the very large size of Web data in today's websites, as well as the well-known drawbacks of being time consuming and less reusable. According to Stumme, Hotho and Berendt, it is impossible to manually discover the meaning of all Web-pages and their usage for a large scale website [10]. Automatic construction of ontologies, on the other hand, can save time and discover all possible concepts within a website and links between them, and the resultant ontologies are reusable. However, the drawback of this automatic approach is the need to design and implement the learning models which can only be done by professionals at the beginning. Therefore, the trade-off between the two approaches to ontology construction needs to be considered and evaluated for a given website.

This paper presents a novel method to provide better Web-page recommendation based on Web usage and domain knowledge, which is supported by three new knowledge representation models and a set of Web-page recommendation strategies. The first model is an ontology-based model that represents the domain knowledge of a website. The construction of this model is semi-automated so that the development efforts from developers can be reduced. The second model is a semantic network that represents domain knowledge, whose construction can be fully automated. This model can be easily incorporated into a Web-page recommendation process because of this fully automated feature. The third model is a conceptual prediction model, which is a navigation network of domain terms based on the frequently viewed Web-pages and represents the integrated Web usage and domain knowledge for supporting Web-page prediction. The construction of this model can be fully automated. The recommendation strategies make use of the domain knowledge and the prediction model through two of the three models to predict the next pages with probabilities for a given Web user based on his or her current Web-page navigation state. To a great extent, this new method has automated the knowledge base construction and alleviated the new-page problem as mentioned above. This method yields better performance compared with the existing Web usage based Web-page recommendation systems.

This paper is structured as follows: Section 2 briefs the related work. Section 3 presents the first model, i.e. an ontology-based domain knowledge model. Section 4 describes the second model, i.e. a semantic network of domain terms. Section 5 presents the third model, i.e. a conceptual prediction model. For each of the models presented in Sections 3-5, the corresponding queries that are used to retrieve semantic information from the knowledge models have been presented. Section 6 presents a set of recommendation strategies based on the queries to make semantic-enhanced Web-page recommendations. Section 7 presents the experiments to compare the performance of the proposed models, algorithms and strategies, along with the experimental results analysis. Section 8 concludes this paper and highlights some further work.

## 2 Related Work

We roughly classify the research work related to Web-page recommendation into the following two categories.

### 2.1 Traditional Approaches that use Sequence Learning Models

In applying sequence learning models to Web-page recommendation, association rules and probabilistic models have been commonly used. Some models, such as sequential modelling, have shown their significant effectiveness in recommendation generation [2]. In order to model the transitions between different Web-pages in Web sessions, Markov models and tree-based structures are strong candidates [2], [11]–[14]. Some surveys [15], [16] have shown that tree-based algorithms, particularly Pre-Order Linked WAP-Tree Mining (PLWAP-Mine for short) [13], are outstanding in supporting Web-page recommendation, compared with other sequence mining algorithms. Furthermore, the integration of PLWAP-Mine and the higher-order Markov model [12] can significantly enhance mining performance [17].

### 2.2 Semantic-Enhanced Approaches

The semantic-enhanced approaches integrate semantic information into Web-page recommendation models. By making use of the ontology of websites, Web-page recommendation can be enriched and improved significantly in the systems [18], [19]. In the systems, a domain ontology is often useful for clustering documents, classifying pages or searching subjects. A domain ontology can be obtained by manual or automatic construction approaches, for example, ontologies have been developed for distance learning courses [20], course content [21], personalized e-learning [22], contracts [23], and software [24]. Depending on the domain of interest in the system, we can reuse some existing ontologies or build a new ontology, and then integrate it with Web mining. For example, ontology concepts are used to semantically enhance Web logs in a Web personalization system [25]. In this system, an ontology is built with the concepts extracted from the documents, so that the documents can be clustered based on the similarity measure of the ontology concepts. Then, usage data is integrated with the ontology in order to produce semantically enhanced navigational patterns. Subsequently, the system can make recommendations, depending on the input patterns semantically matched with the produced navigational patterns. Liang Wei and Song Lei [18] employ ontology to represent a website's domain knowledge using the concepts and significant terms extracted from documents. They generate online recommendations by semantically matching and searching for frequent pages discovered from the Web usage mining process. This approach achieves higher precision rates, coverage rates and matching rates.

On the other hand, by mapping Web-pages to domain concepts in a particular semantic model, the recommender

system can reason what Web-pages are about, and then make more accurate Web-page recommendations [7], [8]. Alternatively, since Web access sequences can be converted into sequences of ontology instances, Web-page recommendation can be made by ontology reasoning [6], [9]. In these studies, the Web usage mining algorithms find the frequent navigation paths in terms of ontology instances rather than normal Web-page sequences. Generally, ontology has helped to organize knowledge bases systematically and allows systems to operate effectively.

## 3 Domain Ontology of a Website for Web-page Recommendation

In the context of Web-page recommendation, the input data is Web logs that record user sessions on a daily basis. The user sessions include information about users' Web-page navigation activities. Each Web-page has a title, which contains the keywords that embrace the semantics of the Web-page. Based on these facts, we aim to discover domain knowledge from the titles of visited Web-pages at a website and represent the discovered knowledge in a domain ontology to support effective Web-page recommendation.

A domain ontology is defined as a conceptual model that specifies the terms and relationships between them explicitly and formally, which in turn represent the domain knowledge for a specific domain [26]. The three main components are listed as follows [21]:

1) Domain terms (concepts),
2) Relationships between the terms (concepts), and
3) Features of the terms and relationships.

Ontologies are often implemented in a logic-based language, such as OWL/RDF, to become understandable to software agents or software systems. Therefore, ontology-based knowledge representation allows sharing and interchanging semantic information among Web systems over the Internet. It also enables the reuse of the domain knowledge, and reasoning the semantics of Web-pages from the existing facts [27]. Furthermore, ontological representation of discovered knowledge from different sources can be easily integrated to support Web-page recommendation effectively.

Depending on the purposes of ontologies, they can be designed as domain conceptualizations of various degrees of formality and can be in the form of concept schemes, taxonomies, conceptual data models, or general logical theories [28]. In this section, we will construct a conceptual data model as a domain ontology for a given website. Since this ontology is used to support Web-page recommendation, we take a Web-page as a unit and assume each page title is well defined to represent key information about the content of the page. The rationale behind this assumption can be seen from two aspects. One aspect is that a Web-page contains a collection of objects (represented by HTML tags) documented in metadata, which is data about data. Metadata embraces the core elements of title, meaning, descriptive context, structure, and overall context of a Web-page. By analysing the metadata, such as Web-page title, the meaning of a Web-page can be understood and captured. The second aspect is from the professional practice in Web development. In well-designed Web-pages, the TITLE tag should contain the meaningful keywords which are relatively short and attractive to support Web search or crawling. In practice, the terms in page titles are usually given higher weights by search engines, such as Google [29], [30]. Consequently, professional website developers need to define the Web-page titles very seriously because they want their Web-pages to be correctly identified during Web search or crawling and use the Web-page titles to convey accurate information about the Web-page. Because of these facts, we use the Web-page titles as clues to represent the domain knowledge of a website. It implies that although there are numerous models for extracting topics of Web-pages, making use of Web-page titles is simple and easy to implement.

TABLE 1
Sample MS Web Dataset

| Page | Title | Path |
|---|---|---|
| $d_1$ | MS Word | /msword |
| $d_2$ | MS Word Support | /mswordsupport |
| $d_3$ | MS Access | /msaccess |
| $d_4$ | MS Access Support | /msaccesssupport |
| $d_5$ | MS in Education | /education |
| $d_6$ | Visual Fox Pro Support | /vfoxprosupport |

This section now presents a procedure for constructing the domain ontology using the Microsoft (MS) website (www.microsoft.com) as an example. The dataset was downloaded from http://kdd.ics.uci.edu/databases/msweb/msweb.html. The ontology will be constructed based on the titles of visited Web-pages so that it is the domain knowledge perceived by users. Queries are then provided based on this domain ontology.

### 3.1 Domain Ontology Construction

There are three steps in the procedure for constructing the domain ontology.

#### 3.1.1 Step 1: Collect the terms

In order to collect the terms, we will: (i) collect the Web log file from the Web server of the website for a period of time (at least seven days), (ii) run a pre-processing unit to analyse the Web log file and produce a list of URLs of Web-pages that were accessed by users, (iii) run a software agent to crawl all the Web-pages in the URL list to extract the titles, and (iv) apply an algorithm to extract terms from the retrieved titles, i.e., single tokens are extracted first by removing stop words from the titles, some single tokens are then combined into composite terms if these single terms often occur at the same time and there is never any token appears between these tokens, and the remaining single tokens will become single word terms.

Using the MS Web dataset, we obtain the Web-page titles and paths. A sample dataset is shown in Table 1.

Given this dataset, the extracted terms, for the sample (Table 1), might be "MS", "Word", "Access", "Support", "Education", "Visual", and "Fox_Pro". Based on the extracted terms, we can generalize them to domain concepts in Step 2.

#### 3.1.2 Step 2: Define the concepts

It is possible for some extracted terms to share the same features, so it is better for them to be instances of a concept, rather than standalone concepts. In this step, the domain concepts will be defined for the given website based on the extracted terms. In this paper, we present the MS website as an example. This website focuses on the application software, such as MS Office, Windows Operating System, and Database. Therefore, the identified domain concepts of this website are *Manufacturer*, *Application*, *Product*, *Category*, *Solution*, *Support*, *News*, *Misc*, *and SemPage*, where the concept *SemPage* refers to the class of Web-pages, and the other concepts refer to the general terms in the MS website.

#### 3.1.3 Step 3: Define taxonomic and non-taxonomic relationships

According to Uschold and Gruminger [31], there are three possible approaches to develop the taxonomic relationships, such as, (4) a top-down development process starts from the most general concepts in the domain and then identifies the subsequent specialization of the general concepts, (5) a bottom-up development process starts from the most specific concepts as the leave nodes in the concept hierarchical structure/tree structure, then groups these most specific concepts into more general concepts, (16) a hybrid development process is the combination of the top-down and bottom-up approaches. We identify the core concepts in the domain first and then generalise and specialise them appropriately.

With the MS website example, we applied a hybrid approach to define taxonomic relationships. We started with the concept *Application* and *Product*. Considering the *consistsOf* relation, which indicates that a concept comprises a number of parts that are also concepts, we particularly have an *application* that may *consist of* some sub-applications. For instance, an *application* software of Office has some sub-applications, including Word, Access, Power Point, etc. Considering the *includes* relation, we have a *product* that may include some sub-products, e.g. Office software *product* includes sub-products such as MS Word, MS Access, and MS Power Point. Considering the *belongsTo* relation, we have a *product* that may *belong to* a certain *category*, e.g. *software*, *hardware*, *entertainment*, or *service*.

The non-taxomonic relationships can be the relationship types used in a relational database except for the relationships between a super-set and a sub-set, such as self-referencing, 1-M and M-N relationships. In the MS website example, the main types of non-taxonomic relationships are listed as below.

1) The 'provides' relation describes the M:N relationship between concept *Manufacturer* and concepts *Product*, *Solution*, *Support*, and *News*. The 'isProvided' relation is the inverse of the 'provides' relation.
2) The 'has' relation describes the M:N relationship between concept *Application* and concepts *Product*, *Solution*, *Support*, and *News*. The 'isAppliedFor' relation is the inverse of the 'has' relation.
3) The 'hasPage' relation describes the M:N relationship between a concept, such as *Application* and *Product* and the concept *SemPage*, e.g. the MS Word *application* has some Web-pages describing its general information and features. The 'isAbout' relation is the inverse of the 'hasPage' relation, which means when we define a page about a certain instance, that instance has the page as its object property value.

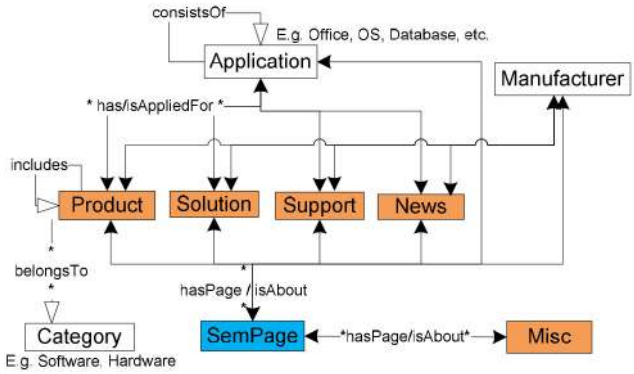


Fig. 1. Domain ontology schema of the MS website.

Combining the taxonomic and non-taxonomic relationships between the concepts, we have a conceptual data model as the ontology schema as shown in Fig. 1.

To formalize this domain knowledge representation model in Description Logics, the domain ontology model of the website is defined as follows.

**Definition 1 (Domain ontology of a website - DomainOnto WP).** *Let $T_{man}$ be a set of domain terms in the given website, $D$ be a set of Web-pages in the given website, $A$ be a set of association relations which are the taxonomic and non-taxonomic relationships in the domain model of the given website, and $\mathcal{A}$ be a set of axioms, e.g., an instantiation axiom assigning an instance to a class, an assertion axiom assigning two instances by means of a property, a domain axiom for a property and a class, and a range axiom for a property and a class. The domain ontology model of the website is defined as a 4-tupes: $O_{man}$: $=< T_{man}, D, A, \mathcal{A} >$, where*

$T_{man} \doteq DomainTerms \equiv Manufacturer \sqcup Application \sqcup Product \sqcup Category \sqcup Solution \sqcup Support \sqcup News \sqcup Misc$,

$D \doteq SemPage$, and

$A \doteq consistsOf \sqcup includes \sqcup belongsTo \sqcup provides \sqcup isProvided \sqcup has \sqcup isAppliedFor \sqcup hasPage \sqcup isAbout$.

The knowledge base of the model is described as follows:

$\exists consistsOf.Application \sqsubseteq Application$,

$\exists includes.Product \sqsubseteq Product$,

$Product \sqsubseteq \exists belongsTo.Category$,

$Manufacturer \sqsubseteq \exists provides.(Product \sqcup Solution \sqcup Support \sqcup News)$,

$(Product \sqcap Solution \sqcap Support \sqcap News) \sqsubseteq \bot$,

$isProvided \equiv provides^{-}$,

$Application \sqsubseteq \exists has.(Product \sqcup Solution \sqcup Support \sqcup News)$,

$isAppliedFor \equiv has^{-}$,

$SemPage \sqsubseteq \exists isAbout.(Manufacturer \sqcup Application \sqcup Product \sqcup Solution \sqcup Support \sqcup News \sqcup Misc)$, and

$hasPage \equiv isAbout^{-}$.

This domain ontology is constructed at three levels: 1) *General level*, which holds the concepts that present the general domain terms of Web-pages and relationship definition sets; 2) *Specific level*, which holds the specific domain terms corresponding to the domain concepts, e.g. terms "Database" and "Office" are the instances of concept *Application*, and the relationships between terms; 3) *Web-page level*, which holds all the Web-pages within the given website, and the association relationships between Web-pages and terms. The general level is presented as an ontology schema, and the specific and Web-page levels are presented as ontology instances. Such an ontology model supports modular development, scalability and reusability at different levels. The general terms within a domain are usually stable and have little changes over the time while the specific terms can increase frequently with the evolution of the Web-page. For instances, when a new Web-page is generated in the site, its title can very likely contain specific terms that are part of existing general terms, but not in the domain ontology. With this ontology structure, these specific terms can be easily added into the ontology at the specific level and the Web-page can be included easily at the Web-page level.

The association relations between the concept *SemPage* and other domain concepts allow the machine to interpret Web-pages or identify what Web-pages are about. However, one problem is how to assign numerous Web-pages to domain terms appropriately. The clue is keywords existing in Web-page titles. Hence, each term instance needs to be specified by relevant keywords. By matching keywords in terms and Web-page titles, the system can automatically map the Web-pages with respect to the domain terms.

This domain ontology, namely **DomainOntoWP**, is implemented using OWL in Protégé. With the help of OWL, we can perform the following queries for the use in the later recommendation process.

### 3.2 Queries

To query for the domain terms (topic) of a given Web-page $d \in D$ at the MS website, we can retrieve concept instances that are associated with the *SemPage* instance $d$ via the 'isAbout' object property. We refer to this query as $\boldsymbol{Topic_{man}(d)}$. Based on Definition 1, this query is expressed in Description Logics as $q_1(x)$ :- *DomainTerms*($x$), *isAbout*($d$, $x$).

In addition, to query for Web-pages of a given domain term $t \in T_{man}$ in the MS website, we can retrieve *SemPage* instances which are mapped to the concept instance $t$ via the 'hasPage' object property. We refer to this query as $\boldsymbol{Page_{man}(t)}$. Based on Definition 1, this query is expressed in Description Logics as $q_2(x)$ :- *SemPage*($x$), *hasPage*($t$, $x$).

## 4 Semantic Network of a Website for Web-page Recommendation

In this section, we will present the second model, i.e. a new semantic network of a website, which is a kind of knowledge map which represents domain terms, Web-pages, and relations including the collocations of domain terms, and the associations between domain terms and Web-pages. First, we collect the domain terms from the Web-page titles based on the assumption that a well-designed Web-page should have an informative title; then we extract the relations between these terms from the following two aspects: (i) the collocations of terms which are determined by the co-occurrence relations of terms in Web-page titles; and (ii) the associations between terms and Web-pages. In addition, the domain terms and co-occurrence relations are weighted to provide a rough indication of how much these terms are associated with each other semantically. Based on the relations between the terms and Web-pages, we can infer how closely the Web-pages are semantically related to each other. Using this model, we can query about the relations between terms and Web-pages, such as the relevant pages for a given page, the key terms for a given page, and the pages for given terms, to infer the semantics of Web-pages to achieve semantic-enhanced Web-page recommendations. This semantic network is referred to as TermNetWP for the convenience in the explanation in this section and the comparisons in Section 7.

### 4.1 TermNetWP

#### 4.1.1 Definitions of TermNetWP

**Definition 2.** *Let $T_{auto} = \{t_i : 1 \le i \le p\}$ be a set of domain terms extracted from Web-page titles, and $D = \{d_j : 1 \le j \le q\}$ be a set of the Web-pages, each page $d_j$ has a sequence of domain terms $X_j = t_1 t_2 \dots t_n$, $t_k \in T_{auto}$ for $1 \le k \le n$ (a domain term may be duplicated in the sequence), which is extracted from the title of that page. Given a Web-page $d_j$ and a domain term $t_i$, $t_i$ is a domain term of $d_j$, as denoted as $t_i \tilde{\in} d_j$, if $t_i \in \{t_1, t_2, \dots, t_n\}$.*

**Definition 3.** *By Definition 2, given the Web-page set $D$ and a term $t$, we define $tf(t, D)$ as the number of occurrences of $t$ over $D$, i.e. the total number of times that $t$ occurs in each title of $d \in D$. Given $TS = \{X_j : 1 \le j \le q\}$ being a set of domain term sequences, and a pair of terms $(t_i, t_j)$, $t_i$, $t_j \in T_{auto}$, we define $\omega\ (t_i, t_j)$ as the number of times that $t_i$ is followed by $t_j$ in TS, and there is no term between them.*

**Definition 4 (Semantic network of Web-pages, *TermNetWP*).** *By Definitions 2 and 3, the semantic network of Web-pages, namely TermNetWP, is defined as a 4-tuples: $O_{auto}$: $=< T, L, D, R >$, where*
*$T = \{(t, f) : t \in T_{auto} \wedge f = tf(t, D) > 0\}$ is a set of domain terms and corresponding occurrences,*
*$L = \{(t_x, t_y, w_{xy}) : t_x, t_y \in T_{auto} \wedge (t_x t_y) \subseteq X_j \wedge X_j \in TS \wedge w_{xy} = \omega(t_x, t_y) > 0\}$ is a set of associations between $t_x$ and $t_y$ (with weight $w_{xy}$), and*
*$R = \{(t, d) : t \tilde{\in} d \wedge t \in T_{auto} \wedge d \in D\}$ is a set of relations between domain term $t$ and Web-page $d$, that is, term $t$ belongs to the title of page $d$.*

#### 4.1.2 Schema of TermNetWP

The schema of TermNetWP is shown in Fig. 2, where class *Instance* defines a domain term, i.e. $t \in T_{auto}$, that has two data type properties, which are *Name* and *iOccur*, and one *WPage* object property. The *iOccur* property refers to the number of occurrences of the term in the set of Web-page titles. Class *WPage* defines a Web-page, i.e. $d \in D$, with

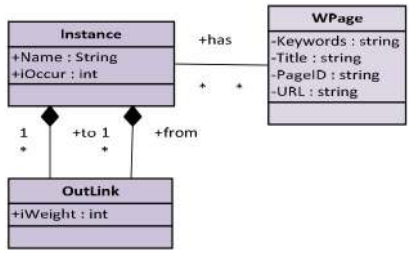


Fig. 2. Schema of TermNetWP.

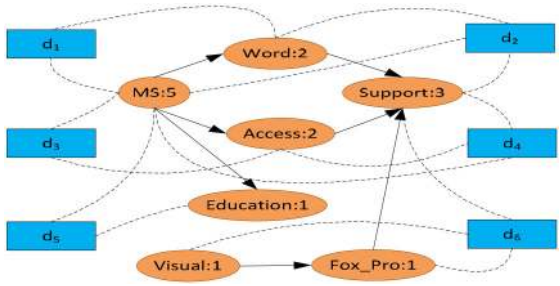


Fig. 3. Illustration of TermNetWP; $t_i$:$k$ = term:occurrence, $d_j$ = page.

properties *Title*, *PageID*, *URL* and *Keywords* in the title. The *Keywords* property is used to store terms in a Web-page title. Classes *Instance* and *WPage* are associated through the 'hasWPage' relationship, i.e. $(t, d) \in R$, from *Instance* to *WPage*, which annotates a constraint: a term instance has one or some Web-pages; and the 'belongto-Instance' relationship, which is the inverse relationship of 'hasWPage', annotates a constraint: a Web-page belongs to one or more term instances.

Moreover, in domain term sequences, each term may have some previous terms and next terms, so an association class *OutLink* is defined to specify the *in-out* relationship between two terms. Class *OutLink* is responsible for connecting *from* one term instance ($t_x$) *to* another term instance ($t_y$), and includes the corresponding connection weight ($iWeight = w_{xy}$). Class *OutLink* therefore involves two object properties: (i) 'from-Instance' referring to one previous term instance, and (ii) 'to-Instance' referring to one next term instance. Correspondingly, class *Instance* also has two object properties: (i) 'hasOutLink' being the inverse of the 'from-Instance' relation, and (ii) 'fromOutLink' being the inverse of the 'to-Instance' relation.

#### 4.1.3 Procedure of Automatically Constructing TermNetWP

In order to construct TermNetWP, we apply the procedure consisting of the following steps:

*Step 1: Collect the titles of visited Web-pages*

In order to collect the titles, we will (i) collect the Web log file from the Web server of the website for a period of time (at least seven days), (ii) run a pre-processing unit to analyse the Web log file and produce a list of URLs of Web-pages that were accessed by users, and (iii) run a software agent to crawl all the Web-pages in the list to extract the titles.

*Step 2: Extract term sequences from the Web-page titles*

We apply the algorithm used in the domain ontology construction to extract the terms from the retrieved titles. The extracted terms are organized in the order as they appear in each title, namely they are collected as term sequences. For example, some term sequences extracted from the titles in the MS website example in Table 1 are: ("MS" "Word"); ("MS" "Word" "Support"); ("MS" "Access"); and ("MS" "Access" "Support"), ("MS" "Education"), and ("Visual" "Fox_Pro" "Support").

*Step 3: Build the semantic network – TermNetWP*

In TermNetWP, each node represents a term in the extracted term sequences and the order of the terms in sequences determines the 'from-Instance' and 'to-Instance' relations of a term between other terms. By scanning all the term sequences extracted from the previous step (Step2), we can build the TermNetWP. For example, a TermNetWP based on the term sequences presented in the Step 2 is simply shown in Fig. 3, where the circles represent terms along with their occurrences and the rectangles represent Web-pages. Each $d_j$ ($j = [1 \ldots 6]$) is mapped to the corresponding terms.

*Step 4: Implement an automatic construction of TermNetWP*

The TermNetWP is implemented in OWL to enable the domain term network to be reused and shared by other parts of a Web-page recommender system. Table 2 shows the algorithm to automatically construct a TermNetWP. The input data is a term sequence collection (*TSC*), in which each record consists of:

TABLE 2
Algorithm to Automatically Construct a TermNetWP

Algorithm: Automatically construct a TermNetWP
*Input*: *TSC* (Term sequence collection)
*Output*: *G* (TermNetWP)
*Process*:
Let $TSC = \{PageID, X = t_1 t_2 \ldots t_m, URL\}$
Initialize *G*
Let *R* = root or the start node of *G*
Let *E* = the end node of *G*
**For each** *PageID* and **each** sequence *X* in *TSC* {
Initialize a *WPage* object identified as *PageID*
**For each** term $t_i \in X$ {
If node $t_i$ is not found in *G*, then
- Initialize an *Instance* object *I* as a node of *G*
- Set $I.Name = t_i$

Else
- Set *I* = the *Instance* object named $t_i$ in *G*

Increase *I.iOccur* by 1
If (*i*==0) then
- Initialize an *OutLink* $R$-$t_i$ if not found
- Increase $R$-$t_i.iWeight$ by 1
- Set $R$-$t_i.fromInstance = R$
- Set $R$-$t_i.toInstance = I$

If (*i*>0 & *i*<*m*) then
- Get *preI* = the *Instance* object with name $t_{i-1}$
- Initialize an *OutLink* $t_{i-1}$-$t_i$ if not found
- Increase $t_{i-1}$-$t_i.iWeight$ by 1
- Set $t_{i-1}$-$t_i.toInstance = I$
- Set $t_{i-1}$-$t_i.fromInstance = preI$

If (*i*==*m*) then
- Initialize an *OutLink* $t_i$-*E* if not found
- Increase $t_i$-*E.iWeight* by 1
- Set $t_i$-*E.toInstance* = *E*
- Set $t_i$-*E.fromInstance* = *I*

Set *I.hasWPage* = *PageID*
Add term $t_i$ into *PageID.Keywords*
}
}

1) The PageID of a Web-page $d \in D$;
2) A sequence of terms $X = t_1 t_2 \dots t_m \in TS, m > 0$, extracted from the title of the Web-page; and
3) The URL of the Web-page.

TermNetWP can be used effectively not only to model the term sequences in connection with Web-pages, but also to present the co-occurrence relations of terms in the term sequences based on the following features: (i) it allows a term node to have multiple in-links and/or out-links so we can easily describe the relationships among terms/nodes in the semantic network, i.e. one node might have previous or next nodes; and (ii) it includes the Web-pages whose titles contain the linked terms so that the meaning of Web-pages can be found through these terms by software agents/systems. More importantly, TermNetWP enables reasoning of relationships between terms and Web-pages within a specific domain.

### 4.2 Queries

As mentioned before, TermNetWP represents the knowledge of domain terms, their associations and the linked Web-pages. Based on TermNetWP, we can query: (i) domain terms of a given Web-page, and (ii) Web-pages mapped to a given domain term, as follows.

#### 4.2.1 Query about terms of a given Web-page

To query for domain terms (topic) of a given Web-page $d \in D$, we can retrieve term instances that are associated with the *WPage* instance $d$ via the 'belongto-Instance' object property. We refer to this query as $\boldsymbol{Topic_{auto}(d)}$. In order to ensure that the degree of coverage of the terms in the domain is taken into account in the Web-page recommendation process later, the returned domain terms are sorted in descending order of their occurrence weights. That is because the more times a domain term occurs on Web-pages, the more likely the term has been viewed. Based on Definitions 2-4, the query is described in logics notation as: $Topic_{auto}(d) = (t_1, t_2, \dots, t_s)$, where $d \in D$; $(t_i, d) \in R$, $i = [1 \dots s]$; and $tf(t_i, D) > tf(t_j, D), (i < j \& 1 \leq i, j \leq s)$.

For instance, based on the sample TermNetWP (Fig. 3), for a given Web-page, say $d_2$, $Topic_{auto}(d_2)$ can be used to query for the terms that are related to this Web-page ($d_2$) by retrieving the terms that are linked with this page as {"MS", "Word", "Support"}. Since the terms linked to this page have different occurrence weight values, as 5, 2, and 3, respectively, the terms are presented in descending order of the weight values, as ("MS", "Support", "Word").

#### 4.2.2 Query about pages mapped to a given term

To query for Web-pages of a given domain term $t \in T_{auto}$, we can retrieve *WPage* instances that are mapped to the term instance $t$ via the 'hasWPage' object property. We refer to this query as $\boldsymbol{Page_{auto}(t)}$. In order to ensure that the degree of relevance of retrieved pages to domain term $t$ is taken into account in the Web-page recommendation process later, the returned pages are sorted in ascending order of connection weights between the Web-pages and domain term $t$. A *connection weight* between a Web-page $d \in D$ and domain term $t$ in TermNetWP $O$ is defined as the total of links from/to domain term $t$ to/from the domain terms of Web-page $d$.

**Definition 5 (Connection weight between a page and a domain term).** *Based on Definitions 2–4, the connection weight between a Web-page $d \in D$ and a domain term $t \in T_{auto}$ is defined as: $\eta(d_j, t) = \sum_{k=1, t_k \tilde{\in} d}^{n} \omega(t_k, t) + \omega(t, t_k)$, where $n = |\{t_k : t_k \tilde{\in} d\}|$ is the number of domain terms in the title of page $d$.*

Based on Definitions 2–5, $Page_{auto}(t)$ is described in logics notation as: $Page_{auto}(t) = (d_1, d_2, \dots, d_s)$, where $(t, d_i) \in R$, $i = [1 \dots s]$; and $\eta(d_i, t) < \eta(d_j, t), (i < j \& 1 \leq i, j \leq s)$.

For instance, based on the sample TermNetWP (Fig. 3), for a given domain term, say "Word", $Page_{auto}$("Word") can be used to query for the Web-pages that are mapped to this domain term as $\{d_1, d_2\}$. The connection weights between domain term "Word" and these two pages in the TermNetWP can be calculated using $\eta(d_1$, "Word") and $\eta(d_2$, "Word"), respectively. The connection weight for $d_1$ is 1, and for $d_2$ is 2. Therefore, the returned pages are presented in ascending order of the weight values, as ($d_1$, $d_2$).

Using the two queries, we can find the terms of a given Web-page or the Web-pages for a given term with the degree of relevance based on the automatically constructed semantic network. This would greatly enrich the candidate pool of items to better support the terms prediction process and top-$N$ Web-page recommendation, which will be presented in Section 6.

## 5 Semantic Knowledge Representation Model of Web Usage of a Website for Web-Page Recommendation

In Section 3, we presented a model of knowledge representation, DomainOntoWP, to capture the domain knowledge of a website for supporting Web-page recommendation, while in Section 4, we presented another model of knowledge representation, TermNetWP, to capture the semantics of Web-pages within a website. Although they are efficient for capturing the domain knowledge and semantics of a given website, they are not sufficient on their own for making effective Web-page recommendations. In order to make better Web-page recommendations, we need semantic Web usage knowledge which can be obtained by integrating the domain knowledge model (DomainOntoWP) or the semantic network (TermNetWP) with Web usage knowledge that can be discovered from Web log files using a Web usage mining technique. In this study, we employ an advanced Web usage mining technique, namely PLWAP-Mine, to discover the Web usage knowledge, which is in the form of frequent Web access patterns (FWAP), i.e patterns of frequently visited Web-pages. We integrate FWAP with DomainOntoWP or TermNetWP in order to result in a set of frequently viewed term patterns (**FVTP**), as shown in Fig. 4. This is the semantic knowledge of Web usage of a website.

**Definition 6.** *Let $D$ be a set of Web-pages, $T$ be a set of different domain terms in the titles of the Web-pages, where $T \equiv T_{man}$ if DomainOntoWP is involved, and $T \equiv T_{auto}$ if TermNetWP is involved. Let $P = \{P_1, P_2, \dots, P_n\}$ be a set of FWAP, where each pattern $P_i$ ($i = [1 \dots n]$) contains a sequence of Web-pages, $n$ is the number of the patterns, and $P_i = d_{i1} d_{i2} \dots d_{im}$, $d_{ik} \in D$, $k = [1 \dots m]$, $m$ is the number of Web-pages in the pattern.*

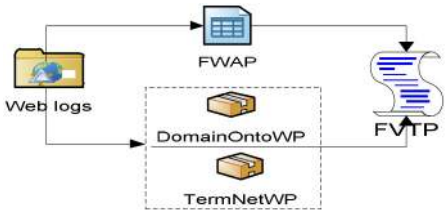


Fig. 4. FVTP discovery.

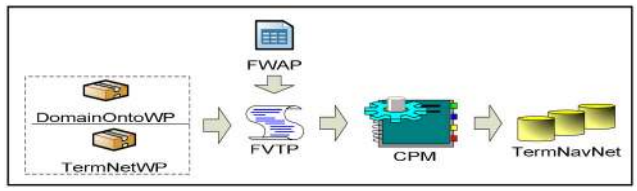


Fig. 5. Term prediction process.

**Definition 7.** *Based on Definition 6, since each page* $(d_{ik}) \subseteq P_i$ $(P_i \in P)$ *contains a set of domain terms* $\tau \subset T$*, we can generate a set of* ***FVTP*** $\boldsymbol{F} = \{t_{i1}t_{i2}\ldots t_{im} : t_{ik} \in T \wedge i = [1\ldots n] \wedge k = [1\ldots m]\}$*, where each domain term pattern* $F = t_{i1}t_{i2}\ldots t_{im}$ *is a sequence of domain terms, in which each domain term* $t_{ik}$ *is a domain term of page* $d_{ik}$ *in* $P_i$*.*

### 5.1 Conceptual Prediction Model (CPM)

In order to obtain the semantic Web usage knowledge that is efficient for semantic-enhanced Web-page recommendation, a conceptual prediction model (**CPM**) is proposed to automatically generate a weighted semantic network of frequently viewed terms with the weight being the probability of the transition between two adjacent terms based on FVTP. We refer to this semantic network as TermNavNet hereafter. Fig. 5 illustrates how CPM acts as a formatter to convert FVTP into TermNavNet.

According to the Markov model [32], a kind of model efficient to represent a collection of navigation records, CPM is developed as a self-contained and compact model. It has two main kinds of elements: (4) state nodes, and (5) the relations between the nodes. One node presents the current state, e.g. current viewed term, and may have some previous state nodes and some next state nodes. By scanning each term pattern $F \in \boldsymbol{F}$, each term becomes a state in the model. There are also two additional states: a start state, $S$, representing the first state of every term pattern; and a final state, $E$, representing the last state of every term pattern. There is a transition corresponding to each pair of terms in a pattern, a transition from the start state $S$ to the first term of a term pattern, and a transition from the last term of a term pattern to the final state $E$. The model is incrementally built by processing the complete collection of FVTP.

**Definition 8.** *Given the domain term pattern set* $\boldsymbol{F}$ *and a domain term* $t_x \in T$*, we define* $\partial_x = |\{F{:}F \in \boldsymbol{F} \wedge (t_x) \subseteq F\}|$ *as the number of occurrences of* $t_x$ *in* $\boldsymbol{F}$*. Given the domain term pattern set* $\boldsymbol{F}$ *and a pair of domain terms* $(t_x, t_y)$*,* $t_x, t_y \in T$*, we define* $\partial_{x,y} = |\{F{:}F \in \boldsymbol{F} \wedge (t_x t_y) \subseteq F\}|$ *as the number of times that* $t_x$ *followed by* $t_y$ *in* $\boldsymbol{F}$ *and there is no term between them. We also define* $\partial_{S,x}$ *as the number of times domain term* $t_x \in T$ *is the first item in a domain term pattern* $F \in \boldsymbol{F}$*, and* $\partial_{x,E}$ *as the number of times a domain term pattern* $F \in \boldsymbol{F}$ *terminates at domain term* $t_x \in T$*. Furthermore, given the domain term pattern set* $\boldsymbol{F}$ *and a transition* $(t_x, t_y, t_z)$*,* $t_x, t_y, t_z \in T$*, we define* $\partial_{x,y,z} = |\{F{:}F \in \boldsymbol{F} \wedge (t_x t_y t_z) \subseteq F\}|$ *as the number of times that* $(t_x, t_y)$ *followed by* $t_z$ *in* $\boldsymbol{F}$ *and there is no term between them.*

The probability of a transition is estimated by the ratio of the number of times the corresponding sequence of states was traversed and the number of times the anchor state occurred. In our system, we take into account first-order and second-order transition probabilities.

**Definition 9 (first-order transition probability).** *Based on Definitions 7 and 8, given a CPM having states* $\{S, t_1, \ldots, t_p, E\}$*, and* $N = |\boldsymbol{F}|$ *is the number of of term patterns in* $\boldsymbol{F}$*, the first-order transition probabilities are estimated according to the following expressions:*

$$\rho_{S,x} = \frac{\partial_{S,x}}{\sum_{y=1}^{N} \partial_{S,y}} \tag{1}$$

*which is the first-order transition probability from the starting state* $S$ *to state* $t_x$*,*

$$\rho_{x,y} = \frac{\partial_{x,y}}{\partial_x} \tag{2}$$

*which is the first-order transition probability from state* $t_x$ *to* $t_y$*,*

$$\rho_{x,E} = \frac{\partial_{x,E}}{\partial_x} \tag{3}$$

*which is the first-order transition probability from state* $t_x$ *to the final state* $E$*.*

**Definition 10 (second-order transition probability).** *Based on Definitions 7 and 8, let* $\rho_{x,y,z}$ *be the second-order transition probability, that is, the probability of the transition* $(t_y, t_z)$ *given that the previous transition that occurred was* $(t_x, t_y)$*. The second-order probabilities are estimated as follows:*

$$\rho_{x,y,z} = \frac{\partial_{x,y,z}}{\partial_{x,y}}. \tag{4}$$

**Definition 11 (Conceptual Prediction Model - CPM).** *Given Definitions 7–10, we formalize the conceptual prediction model as a triple:* $O_p := (N, \Phi, M)$*, where*
$N = \{(t_x, \partial_x) | t_x \in T\}$*: a set of terms along with the corresponding occurrence counts,*
$\Phi = \{(t_x, t_y, \partial_{x,y}, \rho_{x,y}) | t_x, t_y \in T\}$*: a set of transitions from* $t_x$ *to* $t_y$*, along with their transition weights* $(\partial_{x,y})$*, and first-order transition probabilities* $(\rho_{x,y})$*,*
$M = \{(t_x, t_y, t_z, \partial_{x,y,z}, \rho_{x,y,z}) | t_x, t_y, t_z \in T\}$ *: a set of transitions from* $t_x$*,* $t_y$ *to* $t_z$*, along with their transition weights* $(\partial_{x,y,z})$*, and second-order transition probabilities* $(\rho_{x,y,z})$*.*
*IfM is non-empty, the CPM is considered as the second-order conceptual prediction model, otherwise the first-order conceptual prediction model.*

### 5.2 Schema of CPM

In order to automatically construct TermNavNet for a given FVTP, we design the schema of CPM as an ontology schema and implement this schema in the formal ontology language OWL. Its schema consists of classes *cNode*

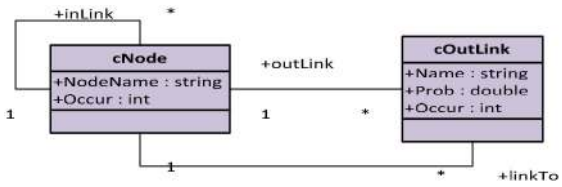


Fig. 6. Schema of conceptual prediction model.

and *cOutLink*, and relationship properties between them, namely *inLink*, o*utLink* and *linkTo* as shown in Fig. 6, where *cNode* and *cOutLink* defines the current state node and the association from the current state node to a next state node, respectively.

The class *cNode* has two object properties *inLink* and *outLink* refering to *cNode* and *cOutLink*, respectively. The number of occurrence of each *cNode* object is represented by *Occur*, i.e. $\partial_x$. *inLink* represents an association from a previous state node, e.g. a previous viewed term, to the state node it belongs to. *cOutLink* represents an association from the state node to one next state node with a transition probability *Prob*, e.g. $\rho_{x,y}$.

### 5.3 Automatic Construction of TermNavNet using CPM

Given a set of frequently viewed term patterns, namely FVTP, we construct TermNavNet by populating the CPM schema with FVTP. An algorithm for accomplishing this task is shown in Table 3.

The transition probabilities in the *cOutLinks* can be updated based on the first-order or second-order probability formula, i.e. (1–3) or (4) depending on the applied CPM's order. As a result, we can obtain a 1st or 2nd-order TermNavNet by using the 1st or 2nd-order CPM, respectively.

### 5.4 Queries

In the context of Web-page recommendation, we might need to query the next viewed terms for a given current viewed term *curT* and a previous viewed term *preT*. We can easily achieve this based on TermNavNet by retrieving *cOutLink* instances that are associated with which *cNode* instance satisfies the following conditions: (i) named *curT*, and (ii) has an *inLink* being *preT*. The second-order CPM is applied to predict the next viewed terms in this query. In order to obtain most frequent terms for later Web-page recommendation, we sort the list of return terms in descending order of transition probabilities assigned in the *cOutLink* instances.

Based on Definition 11, this query is described in logics notation as: $\boldsymbol{RecDTerm}(t_x, t_y) = (t_1, t_2, \ldots, t_s)$, where $t_y \in T$: the current term, $t_x \in T$: the previous term, $(t_x, t_y, t_i)$ is a transition in $M, t_i \in T, i = [1 \ldots s]$, and $\rho_{x,y,i} > \rho_{x,y,j}$, $(i < j \& 1 \leq i, j \leq s)$.

It is noted that when we want to query the next viewed terms for a given currently viewed term *curT* only, the first-order CMP is applied, namely the *inLink* object properties are not taken into account and the first-order transition probabilities are applied instead. Accordingly, the query is expressed as $\boldsymbol{RecDTerm}(t_x) = (t_1, t_2, \ldots, t_s)$, where $t_x \in T$: the current term, $(t_x, t_i)$ is a transition in $\Phi$, $t_i \in T$, $i = [1 \ldots s]$, and $\rho_{x,i} > \rho_{x,j}$, $(i < j \& 1 \leq i, j \leq s)$.

TABLE 3
TermNavNet Construction

| Algorithm: Building TermNavNet |
|---|
| *Input*: $\boldsymbol{F}$(FVTP) |
| *Output*: $M$ (TermNavNet) |
| *Process:* |
| Initialize $M$ |
| **For each** $F = t_1 \ldots t_m \in \boldsymbol{F}$ |
| **For each** $t_i \in F$ |
| Initialize *cNode* objects with *NodeName* = $t_i$, $t_{i-1}$, $t_{i+1}$ and *Occur* = 1 if they are not found in $M$ |
| Initialize a *cOutLink* object with *Name* = $t_i\_t_{i+1}$ and *Occur* = 1 if it is not found in $M$ |
| Increase $t_i$.*Occur* and $t_i\_t_{i+1}$.*Occur* if they are found in $M$ |
| $t_i\_t_{i+1}$.*linkTo* = $t_{i+1}$ |
| $t_i$.*outLink* = $t_i\_t_{i+1}$ |
| $t_i$.*inLink* = $t_{i-1}$ |
| Update all objects into $M$ |
| Update transition probabilities in the *cOutLink* objects |
| **Return** $M$ |

## 6 Semantic-enhanced Web-page Recommendation Strategies

In this section, we present four recommendation strategies, that apply the semantic knowledge base of a given website, which includes the domain ontology of Web-pages (DomainOntoWP) or the semantic network of Web-pages (TermNetWP) and the weighted semantic network of frequently viewed terms of Web-pages within the given website (TermNavNet), to make Web-page recommendations. These recommendations are referred to as semantic-enhanced Web-page recommendations.

**Definition 12.** *Let $S = d_1 \ldots d_k$ be a sequence of Web-pages, which have been visited by a user, with $d_k$ being the currently accessed page and $d_{k-1}$ being the previously accessed page; $F = t_1 \ldots t_k \ldots t_m$ be a pattern of terms in FVTP modelled in TermNavNet with $t_k$ being a currently viewed term of $d_k$, $t_{k-1}$ being a previously viewed term of $d_{k-1}$, and $t_{k+1}$ being a predicted next term.*

For a given current Web-page or a combination of the current and previous Web-pages, the next Web-pages could be recommended differently depending on which knowledge representation model and the order of CPM are used. Four Web-page recommendation strategies are proposed as follows:

*R.DO.1st: uses DomainOntoWP and the first-order CPM*

Step 1 builds DomainOntoWP;
Step 2 generates FWAP using PLWAP-Mine;
Step 3 builds FVTP;
Step 4 builds a 1st-TermNavNet given FVTP;
Step 5 identifies a set of currently viewed terms $\{t_k\}$ using query $Topic_{man}(d_k)$ on DomainOntoWP;
Step 6 infers next viewed terms $\{t_{k+1}\}$ given each term in $\{t_k\}$ using query $RecDTerm(t_k)$ on the 1st-order TermNavNet;
Step 7 recommends pages mapped to each term in $\{t_{k+1}\}$ using query $Page_{man}(t_{k+1})$ on DomainOntoWP.

*R.DO.2nd: uses DomainOntoWP and the second-order CPM*

Steps 1, 2, and 3 are similar to the ones in *R.DO.1st*;
Step 4 builds a 2nd-order TermNavNet given FVTP;
Step 5 identifies a set of previously viewed terms $\{t_{k-1}\}$, and

TABLE 4
Web-Page Recommendation Algorithm

| *R.DO.2nd* | *R.TN.2nd* |
|---|---|
| $R.DO.2nd(d_{k-1}, d_k) =$ $\bigcup_{\substack{t_z \epsilon T_z \\ 1 \le z \le \lvert T_z \rvert}} Page_{man}(t_z)$, where $T_x = Topic_{man}(d_{k-1})$, $T_y = Topic_{man}(d_k)$, | $R.TN.2nd(d_{k-1}, d_k) =$ $\bigcup_{\substack{t_z \epsilon T_z \\ 1 \le z \le \lvert T_z \rvert}} Page_{auto}(t_z)$, where $T_x = Topic_{auto}(d_{k-1})$, $T_y = Topic_{auto}(d_k)$, |
| $T_z = \bigcup_{\substack{t_y \in T_y \\ 1 \le y \le \lvert T_y \rvert}} \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x, t_y)$. | |
| *Notes:* if the first-order CPM is used then $d_{k-1}$ *and* $T_x$ *are not involved.* | |

a set of currently viewed terms $\{t_k\}$ using query $Topic_{man}(d)$, $d \in \{d_{k-1}, d_k\}$, on DomainOntoWP;
Step 6 infers next viewed terms $\{t_{k+1}\}$ given each pair $\{t_{k-1}, t_k\}$ using query $RecDTerm(t_{k-1}, t_k)$ on the 2nd-order TermNavNet;
Step 7 is similar to the one in *R.DO.1st*.

*R.TN.1st: uses TermNetWP and the first-order CPM*

Step 1 builds TermNetWP;
Steps 2, 3 and 4 are similar to *R.DO.1st*;
Step 5 identifies a set of currently viewed terms $\{t_k\}$ using query $Topic_{auto}(d_k)$ on TermNetWP;
Step 6 is similar to the one in *R.DO.1st*;
Step 7 recommends pages mapped to each term in $\{t_{k+1}\}$ using query $Page_{auto}(t_{k+1})$ on TermNetWP.

*R.TN.2nd: uses TermNetWP and the second-order CPM*

Step 1 builds TermNetWP;
Step 2, 3 and 4 are similar to *R.DO.2nd*;
Step 5 identifies a set of previously viewed terms $\{t_{k-1}\}$, and a set of currently viewed terms $\{t_k\}$ using quey $Topic_{auto}(d)$, $d \in \{d_{k-1}, d_k\}$, on TermNetWP;
Step 6 is similar to the one in *R.DO.2nd*;
Step 7 is similar to the one in *R.TN.1st*.

Based on the description of queries in Sections 3–5, the four strategies are described in logics notation as shown in Table 4.

## 7 Experimental Evaluation

In order to evaluate the effectiveness of the proposed models of knowledge representation and the recommendation strategies along with the queries, we implement these models, algorithms and strategies to test their performance of Web-page recommendation using a public dataset. In this section, we firstly list the measures for the performance evaluation of Web-page recommendation strategies, and then present the design of the experiments, followed by the comparisons of experimental results.

### 7.1 Performance Evaluation

The performance of Web-page recommendation strategies is measured in terms of two major performance metrics: *Precision* and *Satisfaction* according to Zhou [14]. In order to calculate these two metrics, we introduce two definitions: Support and Web-page recommendation rules, as follows:

**Definition 13 (Support [2]).** *Given a set $\Delta$ of WAS and a set $P = \{P_1, P_2 \dots P_n\}$ of frequent (contiguous) Web access sequences over $\Delta$, the support of each $P_i \in P$ is defined as: $\sigma(P_i) = \frac{|\{S \in \Delta : P_i \subseteq S\}|}{|\Delta|}$, where $S$ is a WAS.*

In the context of Web usage knowledge discovery using PLWAP-Mine as mentioned in Section 5, *Support* is used to remove infrequent Web-pages and discover FWAP from WAS. This is accomplished by setting a Minimum Support (*MinSup*) and using it as a threshold to check WAS. The Web access sequences whose Support values are greater than, or equal to *MinSup* are considered as FWAP. The smaller MinSup is set, the more FWAP are discovered.

**Definition 14 (Web-page recommendation rules).** *Let $S = s_1 s_2 \dots s_k s_{k+1} \dots s_n$ ($n \ge 2$) be a WAS. For each prefix sequence $S_{prefix} = s_1 s_2 \dots s_k (k \le n-1)$, a Web-page recommendation rule is defined as a set of recommended Web-pages generated by a Web-page recommendation strategy, denoted as $RR = \{r_1, r_2, \dots, r_M\}$, where $r_i (i = [1 \dots M])$ is a recommended Web-page.*

A Web-page recommendation rule is deemed as *correct*, and/or *satisfied*, or *empty* based on the following conditions:

1) If $s_{k+1} \in RR$, $RR$ is *correct*.
2) If $\exists s_i \in RR (k+1 \le i \le n)$, $RR$ is *satisfied*.
3) If $M = 0$, $RR$ is *empty*.

Given a set of recommendation rules, $R = \{RR_1, RR_2 \dots RR_N\}$, where $RR_i (1 \le i \le N)$ is a recommendation rule, and $|R| = N$ is the total number of recommendation rules in $R$ including *empty* rules, the performance of Web-page recommendation strategies is measured in terms of two major performance metrics: *Precision* and *Satisfaction* according to Zhou [14]. The precision is useful to measure how probable a user will access one of the recommended Web-pages. Besides, we also need to consider if a user accesses one of the recommended Web-pages in the near future. Actually, the next page accessed by a user may not be the target page that user wants. In many cases, a user has to access a few intermediate pages before reaching the target page. Hence, the satisfaction is necessary to give the precision that the recommended pages will be accessed in the near future. Particularly, the Precision and Satisfaction can be defined as follows.

**Definition 15 (Precision).** *Let $R_c$ be the sub-set of $R$, which consists of all correct recommendation rules. The Web-page recommendation precision is defined as:*

$$precision = \frac{|R_c|}{|R|}. \quad (5)$$

**Definition 16 (Satisfaction).** *Let $R_s$ be the sub-set of $R$, which consists of all satisfied recommendation rules. The satisfaction for Web-page recommendation is defined as:*

$$satisfaction = \frac{|R_s|}{|R|}. \quad (6)$$

### 7.2 Design of Experimental Cases

We aim to compare the performance of different techniques ranging from the traditional Web-page recommendation approach, such as PLWAP-Mine; through the

semantic-enhanced approach based on a domain ontology of Web-pages (DomainOntoWP), which is constructed manually based on the developer's knowledge, and the DomainOntoWP-based semantic Web usage knowledge (TermNavNet); to the semantic-enhanced approach based on a semantic network of Web-pages (TermNetWP), which is automatically constructed based on the Web usage dataset, and the TermNetWP-based semantic Web usage knowledge (TermNavNet).

Considering the new models of knowledge representation, the queries and the recommendation strategies, we design five experimental cases as follows:

*Case 1 (R.PLWAP).* Set the threshold of the traditional Web-page recommendation approach using the best Web usage mining (WUM) algorithm, i.e. PLWAP-Mine .()()[16]. The Web-page recommendations are generated based on the PLWAP-Mine algorithm. We refer to this case as the base case.

*Case 2 (R.DO.1st).* Test the effectiveness of the semantic-enhanced Web-page recommendation by integrating the domain ontology (DomainOntoWP) with TermNavNet using the first-order CPM. The recommendation strategy (R.DO.1st) is used.

*Case 3 (R.DO.2nd).* Test the effectiveness of the semantic-enhanced Web-page recommendation by integrating the domain ontology (DomainOntoWP) with TermNavNet using the second-order CPM. The recommendation strategy (R.DO.2nd) is used.

*Case 4 (R.TN.1st).* Test the effectiveness of the semantic-enhanced Web-page recommendation by integrating the semantic network of Web-pages (TermNetWP) with TermNavNet using the first-order CPM. The recommendation strategy (R.TN.1st) is used.

*Case 5 (R.TN.2nd).* Test the effectiveness of the semantic-enhanced Web-page recommendation by integrating the semantic network of Web-pages (TermNetWP) with TermNavNet using the second-order CPM. The recommendation strategy (R.TN.2nd) is used.

### 7.3 Training and Testing Data

The Microsoft (MS) website (www.microsoft.com) mentioned in Section 3 was used to run the experimental cases. The dataset was downloaded from http://kdd.ics.uci.edu/databases/msweb/msweb.html. In the dataset, users are identified as numbers and 294 access Web-pages are identified by their titles and URLs. Therefore, this dataset is suitable for the experiments. The dataset is divided into two sub-sets, one for training and one for testing. The two sub-sets are pre-processed in the format of WAS. The training set has 32711 records, and the testing set has 5000 records. The average length of WAS in both sub-sets is 3.

### 7.4 Implementation of Experimental Cases

All five experimental cases are implemented in Java in conjuction with Protégé based on the new models, algorithms and strategies, and the competing recommendation approach (PLWAP-Mine), and run in a Intel Core i5-460M, 2.53 GHz Windows 7 machine with 4GB of RAM.

Two important parameters, namely MinSup, which is the threshold of support values for a Web access sequence to be qualified as a frequent Web access pattern, and recommendation length, which is the number of next recommended Web-pages, might have significant impact on the performance of Web-page recommendation. It was observed on the used dataset that the size of discovered FWAP is too big to run the algorithms when MinSup is lower than 0.3%, and the size of FWAP becomes too small to test the algorithms when MinSup is higher than 1%. Hence, in these experimental cases, we choose a number of MinSup values for each case ranging from 0.3% to 1.0% to control the quantities of FWAP that are discovered from the Web usage mining process, and to guarantee that the amount of generated information is enough to test the proposed models. Regarding the recommendation length, we pre-define the recommendation length to make the system only provide the top-$N$ recommended Web-pages that have higher prediction probabilities. The recommended pages with lower prediction probabilities are ignored to speed the recommendation processes. The recommendation length is limited to these typical values: 5 and 10 for each experimental case depending on the comparisions in Sub-section F.

TABLE 5
Algorithm of Calculating the Evaluation Measures

| Algorithm: Performance evaluation |
|---|
| *Input:* |
| *WASD* (Web access sequence dataset) |
| *MinSup* (the minimum support threshold) |
| *N* (the recommendation length) |
| $O_1$ (TermNavNet) |
| $O_2$ (DomainOntoWP) (or TermNetWP) |
| *Output:* |
| Precision |
| Satisfaction |
| *Process:* |
| **For each** sequence $S_i$ in *WASD*: |
| 1: $S_i = s_1 s_2 \ldots s_k s_{k+1} \ldots s_n$ |
| 2: For each $k \in [1..n\text{-}1]$: |
| a. Set $curP = s_k$ |
| b. Set $preP = s_{k\text{-}1}$ |
| c. Given the parameters ($curP$, $preP$, $O_1$, $O_2$), generate the top-$N$ recommended pages $recP=\{r_1, r_2, \ldots, r_M\}$ using one Web-page recommendation strategy with respect to a certain experimental case |
| d. If $s_{k+1} \in recP$, then increase the number of *correct* recommendation rules $/R_c/$ by 1 |
| e. If $\exists s_i \in recP$ $(k+1 \le i \le n)$, then increase the number of *satisfied* recommendation rules $/R_s/$ by 1 |
| f. Increase the number of recommendation rules $/R/$ |
| Return: the *precision* and *satisfaction*, computed using Equations (5) and (6), respectively |
| *Notes*: in step 2 for Case 1, PLWAP-Mine is used to make Web-page recommendations (refer to the paper [16]) |

### 7.5 Calculation of Evaluation Measures

We develop an algorithm to calculate the performance evaluation measures for all experimental cases as depicted in Table 5.

### 7.6 Comparisons of Experimental Results

We present three sets of comparisons. The first set is the comparisons of experimental results of Cases 2 and 3 against the base Case 1 to validate the effectiveness of semantics brought in by the models based on a manually

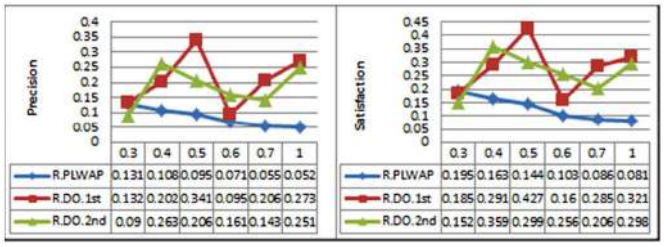


Fig. 7. Results for experimental cases 1, 2, and 3 (Rec.Len = 10).

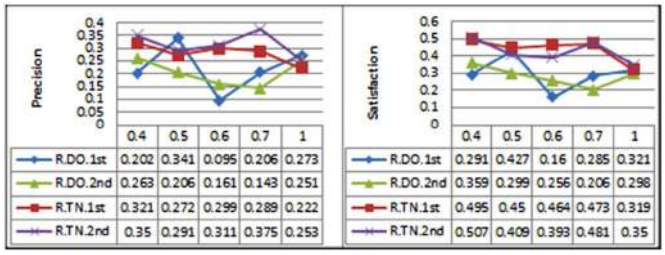


Fig. 8. Results for experimental cases 2–5 (Rec.Len = 10).

constructed domain ontology (DomainOntoWP). The second set is the comparisons of experimental results of Cases 4 and 5 against Cases 2 and 3, respectively, to compare the effectiveness of semantics brought in by the models based on the automatically constructed semantic network of Web-pages (TermNetWP) against the one from the models based on DomainOntoWP. In these first two set of comparisons, the chosen recommendation length is 10.

The third set is the comparisons of the experimental results of Cases 2-5 against Case 1 to compare the effectiveness of semantics brought in by the various models compared with the base case in which there is no semantics used. In these comparisons, the chosen recommendation length is limited to 5.

### 7.6.1 Comparison of Experimental Results for Cases 1–3

The experimental Cases 1–3 were run with different chosen MinSup values, which are 0.3%, 0.4%, 0.5%, 0.6%, 0.7%, and 1%, and the recommendation length of 10. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the three experimental cases are depicted in Fig. 7, where the horizontal axis represents MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively.

From Fig. 7, we make the following observations:

1) Both Cases 2 (R.DO.1st) and 3 (R.DO.2nd) significantly outperform Case 1 in terms of the both measures for almost all the MinSup values as MinSup = {0.4%, 0.5%, 0.6%, 0.7%, 1.0%}. This indicates that adding semantic information into Web-page recommendation significantly enhances the recommendation results. Case 2 almost performs better than Case 3 as MinSup = {0.3%, 0.5%, 0.7%, 1%}, and peaks at MinSup = 0.5%.
2) Furthermore, when the MinSup increases up to 1%, the performance of Case 1 (using PLWAP-Mine) decreases, while that of Cases 2 and 3 relatively fluctuate and are much higher. This is explained as, when MinSup increases, the number of FWAP decreases, so the Web-page recommendation capability of PLWAP-Mine declines. In contrast, the Web-page recommendation using the domain ontology based model (Cases 2 and 3) does not depend much on the MinSup because Web-pages are semantically enhanced with added terms.
3) Case 2 fluctuates more widely than Case 3, this shows that the first-order prediction (R.DO.1st) is more influenced by MinSup than the second-order prediction (R.DO.2nd), but it is able to reach the highest performance.
4) Although it seems that we can decrease the MinSup in Case 1 to get higher performance, the number of FWAP becomes extremely large and run-time will be too long to finish the algorithms. This is not effective in Case 1. It is better to choose Case 2 or 3 with an appropriate MinSup, e.g. 0.5% or 0.4%, respectively, to obtain the high performance of Web-page recommendation.

Overall, the domain ontology based model can improve significantly the performance of Web-page recommendation. The first-order prediction model may be better than the second-order model when the prediction model is based on the domain ontology of Web-pages.

### 7.6.2 Comparison of Experimental Results for Cases 2–5

The experimental Cases 2–5 were run at different chosen minimum supports (MinSup) which are 0.4%, 0.5%, 0.6%, 0.7%, and 1%, and the recommendation length of 10. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the four experimental cases are depicted in Fig. 8, where the horizontal axis represents the MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively.

From Fig. 8, we provide the following observations:

1) Cases 4 and 5 (using TermNetWP) gain higher performance than Cases 2 and 3 (using DomainOntoWP) for MinSup values of 0.4%, 0.6%, and 0.7%.
2) With the same TermNetWP, the second-order CPM (Case 5) is more effective than the first-order CPM (Case 4) in terms of Precision for all test MinSup values.
3) With the same first-order CPM, the DomainOntoWP based model (Case 2) has precisions lower than the TermNetWP based model (Case 4) as MinSup = {0.4%, 0.6%, 0.7%}.
4) With the same second-order CPM, the precision of TermNetWP based model (Case 5) is always higher than that of the DomainOntoWP based model (Case 3).

In general, the semantic network of Web-page based model better raises the performance of Web-page recommendation than the domain ontology based model. Unlike

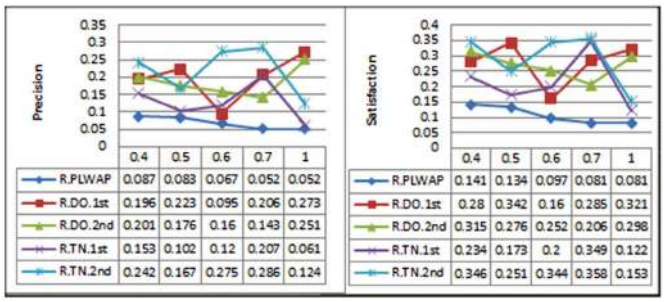


Fig. 9. Results for experimental cases 1–5 (Rec.Len = 5).

the domain ontology based prediction models, the semantic network based second-order CPM is better than the semantic network based first-order CPM.

#### 7.6.3 Comparison of Experimental Results for the all Five Cases

The experimental Cases 1–5 were run at different minimum supports (MinSup) which are 0.4%, 0.5%, 0.6%, 0.7%, and 1%, and the recommendation length of 5. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the five experimental cases are depicted in Fig. 9, where the horizontal axis represents MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively.

From Fig. 9, we make the following observations:

1) The performance of all Cases 1-5 declines as the recommendation length is limited to 5, compared with experimental results found in the two previous sets of comparisons.
2) In terms of both measures, Case 1 (R.PLWAP) is lower than the others for almost all the MinSup values. Case 5 (R.TN.2nd) is mostly higher than the others as the MinSup of 0.4%, 0.6%, and 0.7%.
3) Considering the case of the DomainOntoWP based model, R.DO.1st performs better than R.DO.2nd as the MinSup of 0.5%, 0.7%, and 1%. While, considering the cases of the TermNetWP based model, R.TN.2nd performs better than R.TN.1st as the MinSup of 0.5%, 0.7%, and 1%.
4) Considering the cases of the first-order CPM, the performance of Case 2 is higher than that of Case 4 as the MinSup of 0.4%, 0.5%, and 1%. This is different from the second set of comparisons when the recommendation length is 10.

In overall, the proposed method depends less on the MinSup than PLWAP-Mine, because Web-pages are reasoned based on their semantics. Although an increase in MinSup leads to a decrease in the FWAP, that is the Web-page recommendation capability declines, the proposed method can query for the relevant terms of Web-pages and enrich the pool of recommended Web-pages. As a result, the proposed recommendation models which are based on not only the accessed Web-pages, but also the semantics of Web-pages can outperform the PLWAP-Mine-based recommendation model. Moreover, it has been shown that the second-order prediction model based on TermNetWP is more accurate than the first-order one. However, this is contrary to the DomainOntoWP-based prediction models. Regarding the Web-page recommendation performance of approaches to semantic knowledge representation of Web-pages, the TermNetWP-based models are by large more effective than the DomainOntoWP-based models. In particular, the TermNetWP-based second-order prediction model is the best approach for Web-page recommendation. That is because:

1) TermNetWP allows sorting of terms and Web-pages in the queries, so the most possible items are taken into account in the term prediction and Web-page recommendation processes.
2) TermNetWP fully models terms of Web-pages of user interest, and the relationships between terms and Web-pages. Hence, the interesting Web-pages can be interpreted by the machine.

## 8 Conclusion and Further Study

In conclusion, this paper has presented a new method to offer better Web-page recommendations through semantic enhancement by three new knowledge representation models. Two new models have been proposed for representation of domain knowledge of a website. One is an ontology-based model which can be semi-automatically constructed, namely DomainOntoWP, and the other is a semantic network of Web-pages, which can be automatically constructed, namely TermNetWP. A conceptual prediction model is also proposed to integrate the Web usage and domain knowledge to form a weighted semantic network of frequently viewed terms, namely TermNavNet. A number of Web-page recommendation strategies have been proposed to predict next Web-page requests of users through querying the knowledge bases. The experimental results are promising and are indicative of the usefulness of the proposed models.

Compared with one of the most advanced Web usage mining method, i.e. PLWAP-Mine, the proposed method can substantially enhance the performance of Web-page recommendation in terms of precision and satisfaction. More importnatly, this method is able to alleviate the "new-page" problem mentioned in the introduction because it based on not only the Web usage knowledge, but also the semantics of Web-pages.

For the future work, a key information extraction algorithm will be developed to compare with the term extraction method in this work, and we will perform intense comparisons with the exsiting semantic Web-page recommendation systems.

## References

[1] B. Liu, B. Mobasher, and O. Nasraoui, "Web usage mining," in *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, B. Liu, Ed. Berlin, Germany: Springer-Verlag, 2011, pp. 527–603.
[2] B. Mobasher, "Data mining for web personalization," in *The Adaptive Web*, vol. 4321, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 90–135.
[3] G. Stumme, A. Hotho, and B. Berendt, "Usage mining for and on the Semantic Web," in *Data Mining: Next Generation Challenges and Future Directions*. Menlo Park, CA, USA: AAAI/MIT Press, 2004, pp. 461–480.

[4] H. Dai and B. Mobasher, "Integrating semantic knowledge with web usage mining for personalization," in *Web Mining: Applications and Techniques*, A. Scime, Ed. Hershey, PA, USA: IGI Global, 2005, pp. 205–232.
[5] S. A. Rios and J. D. Velasquez, "Semantic Web usage mining by a concept-based approach for off-line web site enhancements," in *Proc. WI-IAT'08*, Sydney, NSW, Australia, pp. 234–241.
[6] S. Salin and P. Senkul, "Using semantic information for web usage mining based recommendation," in *Proc. 24th ISCIS*, Guzelyurt, Turkey, 2009, pp. 236–241.
[7] A. Bose, K. Beemanapalli, J. Srivastava, and S. Sahar, "Incorporating concept hierarchies into usage mining based recommendations," in *Proc. 8th WebKDD,* Philadelphia, PA, USA, 2006, pp. 110–126.
[8] N. R. Mabroukeh and C. I. Ezeife, "Semantic-rich Markov models for Web prefetching," in *Proc. ICDMW*, Miami, FL, USA, 2009, pp. 465–470.
[9] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Trans. Internet Technol.*, vol. 4, no. 4, pp. 344–377, Nov. 2004.
[10] G. Stumme, A. Hotho, and B. Berendt, "Semantic Web mining: State of the art and future directions," *J. Web Semant.*, vol. 4, no. 2, pp. 124–143, Jun. 2006.
[11] B. Zhou, S. C. Hui, and A. C. M. Fong, "CS-Mine: An efficient WAP-tree mining for Web access patterns," in *Proc. Advanced Web Technologies and Applications*. vol. 3007. Berlin, Germany, 2004, pp. 523–532.
[12] J. Borges and M. Levene, "Generating dynamic higher-order Markov models in Web usage mining," in *Proc. PKDD*, Porto, Portugal, 2005, pp. 34–45.
[13] C. I. Ezeife and Y. Lu, "Mining Web log sequential patterns with position coded pre-order linked WAP-tree," *Data Min. Knowl. Disc.*, vol. 10, no. 1, pp. 5–38, 2005.
[14] B. Zhou, S. C. Hui, and A. C. M. Fong, "Efficient sequential access pattern mining for web recommendations," *Int. J. Knowl.-Based Intell. Eng. Syst.*, vol. 10, no. 2, pp. 155–168, Mar. 2006.
[15] C. Ezeife and Y. Liu, "Fast incremental mining of Web sequential patterns with PLWAP tree," *Data Min. Knowl. Disc.*, vol. 19, no. 3, pp. 376–416, 2009.
[16] T. T. S. Nguyen, H. Lu, T. P. Tran, and J. Lu, "Investigation of sequential pattern mining techniques for Web recommendation," *Int. J. Inform. Decis. Sci.*, vol. 4, no. 4, pp. 293–312, 2012.
[17] S. T. T. Nguyen, "Efficient Web usage mining process for sequential patterns," in *Proc. IIWAS*, Kuala Lumpur, Malaysia, 2009, pp. 465–469.
[18] L. Wei and S. Lei, "Integrated recommender systems based on ontology and usage mining," in *Active Media Technology*, vol. 5820, J. Liu, J. Wu, Y. Yao, and T. Nishida, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 114–125.
[19] A. Loizou and S. Dasmahapatra, "Recommender systems for the semantic Web," in *Proc. ECAI*, Trento, Italy, 2006.
[20] D. Dzemydiene and L. Tankeleviciene, "On the development of domain ontology for distance learning course," in *Proc. 20th EURO Mini Conf. Continuous Optimization Knowledge-Based Technologies,* Neringa, Lithuania, 2008, pp. 474–479.
[21] S. Boyce and C. Pahl, "Developing domain ontologies for course content," *Educ. Technol. Soc.*, vol. 10, no. 3, pp. 275–288, 2007.
[22] J. M. Gascuena, A. Fernandez-Caballero, and P. Gonzalez, "Domain ontology for personalized e-learning in educational systems," in *Proc. 6th IEEE ICALT*, Kerkrade, Netherlands, 2006, pp. 456–458.
[23] Y. Yalan, Z. Jinlong, and Y. Mi, "Ontology modeling for contract: Using OWL to express semantic relations," in *Proc. EDOC'06*, Hong Kong, China, pp. 409–412.
[24] D. Oberle, S. Grimm, and S. Staab, "An ontology for software," in *Handbook on Ontologies*, vol. 2, S. Staab and R. Studer, Eds. Berlin, Germany: Springer, 2009, pp. 383–402.
[25] M. Eirinaki, D. Mavroeidis, G. Tsatsaronis, and M. Vazirgiannis, "Introducing semantics in Web personalization: The role of ontologies," in *Proc. EWMF*, Porto, Portugal, 2006, pp. 147–162.
[26] G. Antoniou and F. V. Harmelen, *A Semantic Web Primer*. Cambridge, MA, USA: MIT Press, 2008.
[27] A. Harth, M. Janik, and S. Staab, "Semantic Web architecture," in *Handbook of Semantic Web Technologies*, J. Domingue, D. Fensel, and J. A. Hendler, Eds. Berlin, Germany: Springer-Verlag, 2011, pp. 43–75.
[28] S. Grimm, A. Abecker, J. Völker, and R. Studer, "Ontologies and the semantic Web," in *Handbook of Semantic Web Technologies*, J. Domingue, D. Fensel, and J. A. Hendler, Eds. Berlin, Germany: Springer, 2011, pp. 507–580.
[29] B. Liu, "Information retrieval and Web search," in *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, B. Liu, Ed. Berlin, Germany: Springer, 2011, pp. 183–236.
[30] Z. Markov and D. T. Larose, "Information retrieval and web search," in *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*, Z. Markov and D. T. Larose, Eds. New Britain, CT, USA: Wiley, 2007, ch. 1, pp. 3–46.
[31] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–36, Jun. 1996.
[32] J. Borges and M. Levene. (May 26, 2004). *A Dynamic Clustering-Based Markov Model for Web Usage Mining,* Technical Report [Online]. Available: http://xxx.arxiv.org/abs/cs.IR/0406032

**Thi Thanh Sang Nguyen** is currently a third year Ph.D. candidate at the Faculty of Engineering and Information Technology (FEIT), the University of Technology, Sydney (UTS), NSW, Australia. Her research topic is semantic web-page recommender system. She received the master's degree in computer engineering from the University of Technology (VNU-HCMC) in 2006. She was a Researcher at the School of Computer Science & Engineering at the International University (VNU-HCMC). She has published five research papers in the field of web mining. Her current research interests include web mining, semantic web, knowledge discovery, and business intelligence.

**Hai Yan Lu** is with the School of Software in FEIT, and a core member of the Decision Systems and e-Service Intelligence (DeSI) Laboratory in the Centre for Quantum Computation & Intelligent Systems (QCIS) at UTS, NSW, Australia. She received the bachelor's and master's degrees from the Harbin Institute of Technology (HIT), China, in 1985 and 1988, respectively, and the Ph.D. degree from UTS in 2002. Her current research interests include heuristic optimization, recommender systems, e-Government, e-Service intelligence, and numerical simulation of electromagnetic devices.

**Jie Lu** is a Head of the School of Software in FEIT, and the Director of the DeSI Laboratory in the Centre for QCIS at UTS, NSW, Australia. Her current research interests include emergency decision-making model, uncertain information processing, early warning systems, recommender systems, and their applications in e-Government, e-Business, and e-Service intelligence. She has published five research books and more than 300 papers in refereed journals and conference proceedings. She serves as Editor-in-Chief for *Knowledge-based Systems* (Elsevier), Editor-in-Chief for *International Journal of Computational Intelligence Systems* (Atlantis/Taylor and Francis), and Editor for the book series on *Intelligent Information Systems* (World Scientific).