SCORE REGION ALGEBRA

A FLEXIBLE FRAMEWORK FOR STRUCTURED INFORMATION RETRIEVAL

Vojkan Mihajlović

CTIT Ph.D. thesis Series No. 06-94

Centre for Telematics and Information Technology (CTIT)

P.O. Box 217 - 7500 AE Enschede - The Netherlands

SIKS Dissertation Series No. 2006-26

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.

ISBN: 90-365-2428-8
ISSN: 1381-36-17 (CTIT Ph.D. thesis Series No. 06-94)
Cover design: Vesna & Marija Smiljanić
Printed by: Wöhrmann Print Service, Zutphen, The Netherlands

SCORE REGION ALGEBRA

A FLEXIBLE FRAMEWORK FOR STRUCTURED INFORMATION RETRIEVAL

DISSERTATION

to obtain
the doctor's degree at the University of Twente,
on the authority of the rector magnificus,
prof. dr. W. H. M. Zijm,
on account of the decision of the graduation committee
to be publicly defended
on Thursday, December 07, 2006 at 13:15

by

Vojkan Mihajlović

born on January 27, 1978
in Niš, Serbia

This dissertation is approved by:

Prof. dr. Peter M. G. Apers (promotor)
Dr. ir. Djoerd Hiemstra (assistant-promotor)

Acknowledgments

The thesis that lies before you is the first book that I wrote in my life. As I am not sure when the next one will come, I would not gamble with the opportunity to acknowledge most of you whose thoughts, actions, and reactions are engraved in me and also the ones that are still willing to enlighten me in the future. Although I cannot mention all of you here, a special place for you will always exist in my heart and in my thoughts.

I start with persons who are responsible for my early education and for encouraging me in my development as a researcher. These are my teachers from primary school "Ratko Vukićević" (Niš): Marina, Živka, Soća, and others, who also gave me many valuable lessons about life. I am also thankful to all my friends from the fourth class with whom I have shared the childhood experiences and good and bad moments for eight years. Many thanks to all of my teachers from the "Bora Stanković" Gymnasium (Niš) and all my classmates from the third class that taught me a lot about human relations and self confidence.

I am in debt to my professors from the Faculty of Electrical Engineering Niš and a small group of students from the back of the 'amphitheater' – Dado, Šaki, Gorča, Milica, Ana, Tijana, Bojana, Vesna, Stojke – who are responsible for helping me to cultivate my professional skills, allowing me to do what I like and to like what I do. Special thanks goes to Slobodanka Djordjević-Kajan for support and encouragement in the later stages and for being my mentor when graduating, as well as to members of CG&GISLAB for their understanding, great work environment, and assistance through my development as a researcher.

I am particularly thankful to Milan Petković for inviting me to a half year "Formula 1 research race" at the University of Twente and for steering my research at that time, and to Willem Jonker for supporting me in the research and for believing in my work as a student, later as a PhD student, and beyond. Also many thanks to Database group members at that time (in 2001) for providing me such a worm and hospitable environment.

Considering my PhD research I am most in debt to my assistant-promotor Djoerd Hiemstra for being my mentor, critic, actuator, and friend, and to my promotor Peter Apers for all punctual and sharp comments and his positivism about my work. They supported me and guided me through my PhD path. Many thanks to Henk Ernst for formal and less formal discussions we had in the last years and for his comments on my thesis. All former and current members of the Database Group at the University of Twente – Maarten, Maurice, Henk, Ling, Rick, Roelof, Mila, Nirvana, Wijnand, Joeri, Henning, Arthur, Ander, Pavel, Harald, Sander, Robin, Robert, Riham – thank you for so many nice moments, chats,

and serious discussions, and for understanding my obstacles with Dutch language. Sandra, Suse, and Ida, thank you for all the help in the tedious paperwork and for awaiting me always with a smile. Many thanks to the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) for sponsoring my research (within the CIRQUID project), to the INitiative for the Evaluation of XML Retrieval (INEX) community and all those productive discussions that we had in Dagstuhl, and especially the CIRQUID team from Centrum voor Wiskunde en Informatica – Nina, Arjen, and Thijs – for many useful advices and great cooperation we had in the past four years.

A great contribution to my development and also in completing this thesis comes from my close friends and family. The support comes from little Andrija and his parents Joe and Ljilja who never lack positive energy and comic stories. Special thanks to 'my architects' Marci and Irenče whose creativity and dynamic spirit always inspires me. Dado and Marina thank you for always being there for me and keeping the joyfulness in all our encounters. I am grateful to Jelena and Boris and 'just married' Tanja and Zoki for all the skating tours, basketball games, and crazy and long discussions that we had on various topics. Many thanks to 'YU community' – Dule, Ljuba, the family Živković, Nataša, Raša, Alek and Jasminka, Boki, Ana, Jelena, Dragon, Tanja T, Novica, Katarina and Bart, and Maksa and Sanja, with whom I shared my work experience, played various sports, and discussed many issues about residing in The Netherlands.

I am thankful to Arthur and Henning, my paranymphs, for the harmonic environment that we had in our office, all brainstormings, jokes, sad and happy moments that we shared in it, and also for our unforgettable camping in Sardegna. Henning thank you for being so open and knowable in our discussions and for being always so full of life.

I also say thanks to my two 'families' in The Netherlands: the family Smiljanić and the family Petković. Thank you 'čika Marko', for all candies and apples, all kind words, life stories, walks, and talks that we shared, and thank you Marina, Marija, and Vesna, for all the games re-awaking the child in me and all unforgettable evening companionships. 'Bata Šimo', thank you for all 'driving lessons' and advices, for all support throughout my work in the computer science area, and for sharing your valuable experience with me. Mihailo, Marko, and 'teta Maco', my gratitude to you for letting me discover myself through some childish and some more serious games, and for enabling the persistence of my research spirit.

My deepest gratitude to my 'Macedonian family' Trajčevski. Thank you Sandričak, 'teta Julo', and 'čika Zorane', for all the warmth and love that you have for me, for letting me feel at home from the first moment, for understanding my ambitions, and for the great support that I had from you during my PhD research. I wish to thank also Nena and Jozo, then Matej, Ivana, and Viksa, and the families Velickoviḱ and Arsovski for open communication and for uplifting my 'Balkan spirit'.

My gratitude to the family Filipović, to my 'Uki' Miodrag for encouraging his 'sestrići' at any moment, and together with my aunt Rada for always believing and strengthening me, to Filip and Tamara for all the plum and grape picking actions we had, for the unforgettable time at the see-side and in our yard in Donji Matejevac, and all the low and high moments we shared in our lives.

Among the people that I want to thank a special place is reserved for my granparents: 'baba' Zorka and Meca, and 'nana' Vuka and 'deda' Mita. Although 'baba' Zorka and 'deda' Mita are now gathering hay and picking fruits together in heaven, they always have a special place in my hart. Thank you all for your eternal positivism and your incredible working energy, and thank you for making me more laborious, more versatile, more spirited, and more comical.

I owe so much gratefulness to my parents Andjica and Tomislav for everything they taught me, for always carrying about me, fighting for my progress, and following me and upholding me in my path. Thank you for enabling me to became the person I am now and for sharpening my sense of reality at every moment in time. A big awe to Vladan, my brother, a person that has the biggest heart in the world. Thank you for being my brother, my norm, my friend, my playmate, my shelterer, my companion. Throughout all the words and silent moments you teach me patience, sympathy, knowledge, comprehension, and many more. You continue to teach me even now and you will always be my big brother.

Most of all, I am grate-grateful to my Zvončica for always helping me to find my happy thought and for teaching me to fly. Thank you for letting me be a part of your life, for showing me the value of true love, what are the really important matters, and what is that one thing that leads me in my journey. Thank you for being with me in every step in my professional life and for feeding me with positive energy that is also enlaced in this thesis.

Finally, I want to thank to Mother Nature for being always so generous and so understanding for me, for tolerating my mistakes and my absurdities and for rewarding me for my kindness, endeavors, and sometimes foolishness. I thank also all her creatures (beside humans) for expanding my views and for leading me to peace and harmony. Thank you also for letting me engrave these letters in your flow of life.

Vojkan Mihajlović
September 2006

Contents

Chapter 1

Introduction

In the origin of every intelligent life form on earth there is curiosity. For humans curiosity is a desire to discover something new, to gather more data, more information, more knowledge. This desire can be clearly seen when looking at the youngest in our population. Small children are always full of questions and look at the world from the perspective of a real researcher: by asking questions, listening, and thinking. They are trying to build something new or to discover the world that surrounds them. They constantly *search* for *information* that would answer questions burning in their mind. Having an *information need* and knowing *where* to find the information, such as from books, from TV programs, from their parents or teachers, the only thing that a child has to find out is *how* to reach the right information. Therefore, he/she could ask parents or teachers a question, or ask a librarian for a book that might provide the answer. This is the *information search* process in early childhood.

Most people retain such an inquisitive mind when they grow up. However, they cannot rely entirely on the knowledge of their parents and teachers or librarians. Furthermore, in the last two decades quite a large portion of the *information sources* moved from paper (books) to digital media (computers), e.g., intranets or the Internet. Nowadays people mainly look for information on the Web. When issuing a question, people still know what they need and where they can find the answer but no longer how. The 'how' part is now mostly left to computer intelligence, usually called *information retrieval systems* or *search engines*, instead of librarians, parents, or teachers. These intelligent information retrieval systems need to search large Web data sources (*databases*) to *retrieve relevant answers.*

Among the innumerable sources of information on the Web, usually termed *documents*, a growing number of documents exists that come with some form of structured annotation, denoting, for example, title, year, author, type, section, or image in a document. These annotated documents are usually termed *structured documents.* At this point we come to the main topic of this thesis: information retrieval in structured documents, that is, *structured information retrieval.*

1.1 Research domain: structured documents

Much research has been done into retrieving relevant documents given simple list-of-term queries, for example, search requests for documents that have structure

Figure 1.1: An XML example: Marked up short story “Selfish Giant” written by Oscar Wilde.

```
<story id="78">
  <title>The Selfish Giant</title>
  <author>Oscar Wilde</author>
  <image src="/images/Garden.jpg">
    <title>Rock garden</title>
  </image>
  <p>Every afternoon, as they were coming from school, the children used to go and play
    in the Giant's garden.</p>
  <p>It was a large lovely garden, with soft green grass.  Here and there over the grass
    stood beautiful flowers like stars, and there were twelve peach-trees that in the spring-
    time broke out into delicate blossoms of pink and pearl, and in the autumn bore rich
    fruit.  The birds sat on the trees and sang so sweetly that the children used to stop their
    games in order to listen to them. "How happy we are here!" they cried to each other.
  </p>
  <p>One day the Giant came back.  He had been to visit his friend the Cornish ogre, and
    had stayed with him for seven years.  After the seven years were over he had said all
    that he had to say, for his conversation was limited, and he determined to return to his
    own castle.  When he arrived he saw the children playing in the garden.</p>

  (...)

  <p>"Who art thou?" Said the Giant, and a strange awe fell on him, and he knelt before
    the little child.</p>
  <p>And the child smiled on the Giant, and said to him, "You let me play once in your
    garden, to-day you shall come with me to my garden, which is Paradise."</p>
  <p>And when the children ran in that afternoon, they found the Giant lying dead under
    the tree, all covered with white blossoms.</p>
</story>
```

can be expressed as a list of two terms “structured documents”. With the rapid growth of data stored in structured format, information retrieval (IR) on structured collections is becoming an important issue. This thesis begins with the characterization of structured documents and with emphasizing the differences between unstructured and structured documents.

The definition of (semi-)structured (markup) languages[1], such as eXtensible Markup Language (XML) [22], Standard Generalized Markup Language (SGML) [54], and HyperText Markup Language (HTML) [174], implies the presence of meta-information about the document content as well as information on the logical organization of a document, besides content. An example structured (XML) document is given in Figure 1.1, where a part of a marked up short story by Os-

[1]These documents (languages) were first denoted as semi-structured as in the beginning they were loosely marked up and had implicit structure [125], and to differentiate from the class of explicitly structured documents and languages used in, e.g., relational database theory [1].

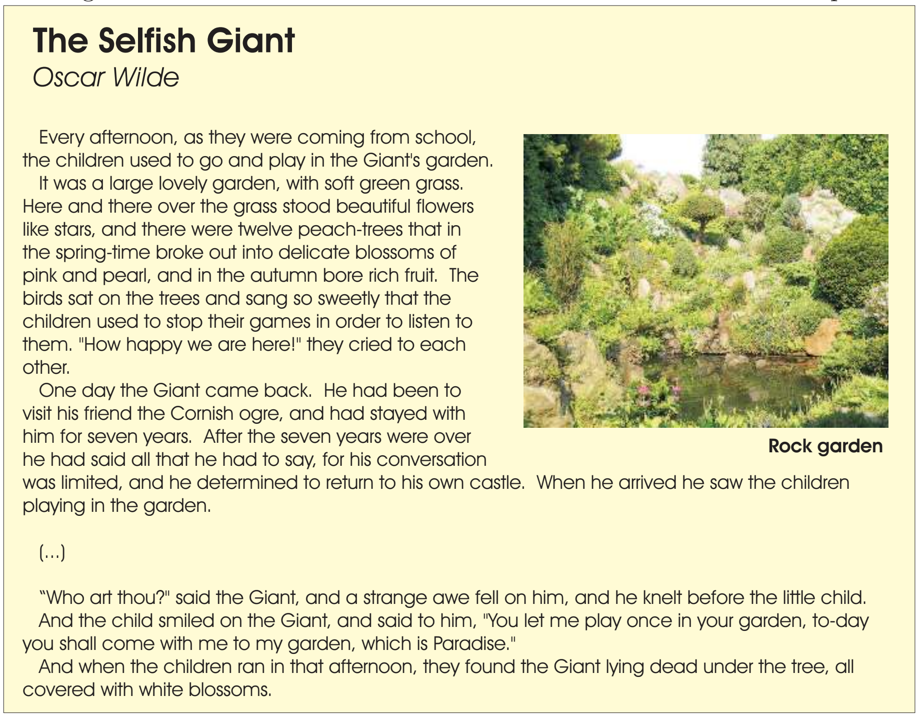


Figure 1.2: Presentation version of the “Selfish Giant” XML example.

The Selfish Giant

Oscar Wilde

Every afternoon, as they were coming from school, the children used to go and play in the Giant's garden.

It was a large lovely garden, with soft green grass. Here and there over the grass stood beautiful flowers like stars, and there were twelve peach-trees that in the spring-time broke out into delicate blossoms of pink and pearl, and in the autumn bore rich fruit. The birds sat on the trees and sang so sweetly that the children used to stop their games in order to listen to them. "How happy we are here!" they cried to each other.

One day the Giant came back. He had been to visit his friend the Cornish ogre, and had stayed with him for seven years. After the seven years were over he had said all that he had to say, for his conversation was limited, and he determined to return to his own castle. When he arrived he saw the children playing in the garden.

Rock garden

(...)

“Who art thou?" said the Giant, and a strange awe fell on him, and he knelt before the little child.

And the child smiled on the Giant, and said to him, "You let me play once in your garden, to-day you shall come with me to my garden, which is Paradise."

And when the children ran in that afternoon, they found the Giant lying dead under the tree, all covered with white blossoms.

car Wilde [219] is presented. As can be seen from the figure, data in structured documents can be classified into two broad categories:

- data that represents information about document structure, content, and layout, i.e., document markup, enclosed in angle brackets ‘<’ and ‘>’: story, title, author, image, etc.
- data that represents content information in structured documents: ‘The Selfish Giant’, ‘Oscar Wilde’, ‘Rock Garden’, etc.

Strictly speaking, any document can be considered to have structure, based on one or more structure-types. For example, flat text (unstructured) documents have a linear order of words, sentences, and paragraphs, and are logically (hierarchically) organized in chapters, sections, subsections, etc. (see Figure 1.2). Similarly, multimedia documents (images and videos) convey spatial and temporal relationships. However, these types of document structure are usually implicit and require a mechanism for their detection.

On the other hand, structured documents have explicit structure expressed through document representation standards, i.e., markup languages. Furthermore, these "structured elements" are used to tag not only the structured organization of documents, but the semantics of a document and its components, as well as document formatting and layout. Thus, the structure (markup) can be used to describe [70]:

1. external document attributes that describe data outside the document content, such as creator name, access rights, and publication information;

2. document layout (e.g., SMIL [25]) used for presenting the document to a user through the output media, such as two-column format, page size A4, and color;

3. document logical structure (e.g., XML, SGML) used to annotate document structure and different types of information it contains, such as title, paragraph, and section;

4. document content (e.g., OWL web ontology language [132]) that describes the high-level meaning of a document – document semantics, such as subject, category, premise, proof, and class.

Thus, structured documents bring more opportunities for modeling information, and also for defining more complex *information retrieval* (*IR*) tasks. Problems encountered in structured information retrieval are discussed in the following section.

1.2 Research problem

The focus of our research is on structured retrieval where we expect (expert) users to be responsible for expressing their structured information need in detail. The aim of the designated retrieval system is to support diverse structured user queries and to model different aspects of user requests, without heavily exploiting the semantic relation between the document markups and content of these marked up document segments.

To discuss the main topic addressed in this thesis we approach the structured retrieval problem from two different perspectives (although other views on the problem are possible): the flat text information retrieval perspective and the database management system perspective. After identifying the problems, a short discussion on IR systems of the future is presented to be used as a guideline for the research explained in this thesis.

1.2.1 Information retrieval

With the rapid growth of data stored in structured format, especially XML, ranked information retrieval from structured collections is becoming a requirement for modern information retrieval systems. Furthermore, users start to be aware that the structured format of documents can be useful for improving the quality of the answers given by retrieval engines. The importance of field search [30, 49, 207] and structured (XML) search [7, 72, 134, 155] is also recognized by many commercial search engines. For instance, Google[2] has "advanced search" that supports a number of field search extensions, such as domain search (*site:*), file type search (*filetype:*), and URL search (*inurl:*). Similarly, the U.S. National Center for Biotechnology Information[3] uses a search engine that searches various medical data sources based on their structure, e.g., users can search the PubMed medical collection based on *subsets* (e.g., AIDS, cancer), *ages* (e.g., infant, child, adult), etc. Furthermore, there are open source search engines such as Indri [207] and Terrier [158], or enterprise search engines such as Panoptic [91], that enable formulation of complex structured queries and their evaluation.

Flat text IR

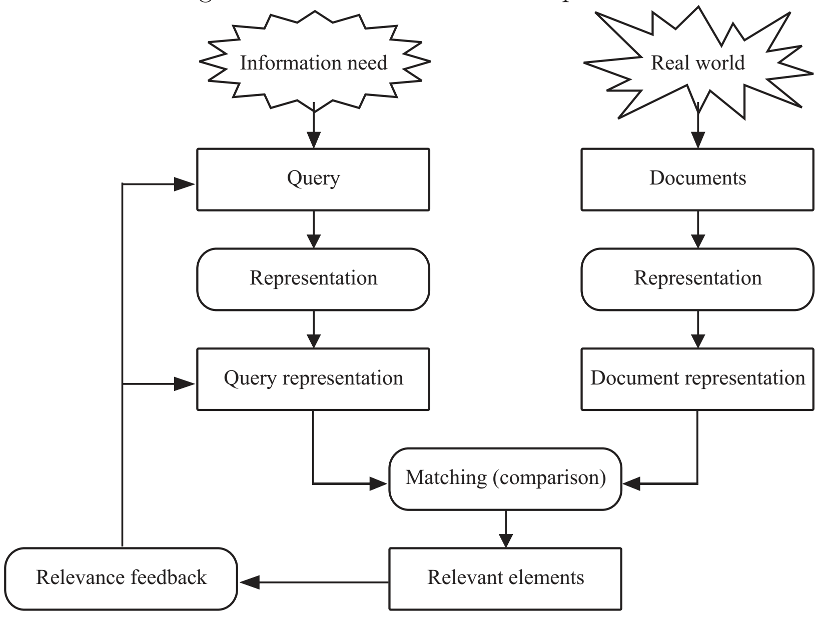
To explain the problems encountered in structured IR we make a distinction between the information retrieval process in structured IR and flat text IR. The basic task of traditional (flat text) information retrieval systems is to perform a search based on a user query, consisting in most cases of a list of query terms. Using a retrieval model (retrieval approach or search method) and data from documents stored using some indexing structure, IR systems should provide the user with a ranked list of answers satisfying his information need. The commonly accepted information retrieval process [48] (recited in [95, 217]) is visualized in Figure 1.3.

To clarify the information retrieval process we explain the terminology that is also used throughout the thesis. Following the conventional IR theory [12, 213] *documents* can be defined as writings that convey information in textual, graphical, and video format. In computer science, documents are often referred to as computer files that are not executable files but contain data that can be used by applications or humans. We use the term *query* to denote both unstructured (list of term) queries as well as structured queries that include structured constraints and search terms. *Query language* denotes the language used for expressing queries.

The user, being able to *formulate* his requests in a specific query language, expresses his information need by a query. The query is then *represented* as a set of query primitives that can be used for the *matching* (*comparison*) process. The real world is *described* in terms of documents, where each document is *represented* (*indexed*) using an internal document representation structure, that can be used for matching. The matching or comparison is then performed on document and query representations (using a retrieval model) and the results, a ranked list of documents/elements, are retrieved. The retrieved answers can be used to further refine the query, automatically or by users. This is represented by the *relevance feedback* loop.

[2]http://www.google.com/.
[3]http://www.ncbi.nlm.nih.gov/.

Figure 1.3: Information retrieval process.



In the information retrieval process, queries are executed using a *retrieval engine* or *retrieval system*. A retrieval system is based on one or more *retrieval models* that define how a *relevance score* is assigned to a document or an element (document component) in the document structure with respect to how well they match a query. A retrieval model is a mathematical framework that defines the comparison (matching) between document and query representations. After determining the relevance score of documents or elements, they are usually *ranked* in descending order of their relevance scores, of which only the highest ranked ones are presented to the user.

Structured IR

The information retrieval process given in Figure 1.3 also holds for structured IR. However, while in traditional (flat text) IR systems queries are represented as a "list of terms" [103], documents as a "bag of words" [95], and where an "inverted index" stores *(term,document)* pairs [213], the components of the structured IR process are far more complex. In structured IR, the user has the opportunity to specify where inside the document he would like to search for information or which part of the document he would like to see as an answer. In other words, richer *query languages*, *retrieval models* that utilize document structure and that are more complex than flat text ones, as well as different *storage* (*indexing*) *structures* have to be formalized, compared to standard (flat text) IR systems. Therefore, a more precise specification of the search intentions of a user, in terms of a more precise definition of the search space (using the structure of documents), needs to be supported.

In this thesis the focus is on structured documents where markup is used to model logical structure and to partially annotate document content. These documents, most frequently specified in XML or SGML format, are widely used on the Web. They also provide useful structured information for the information retrieval task. Furthermore, we use semantically tagged document content that is useful for IR, although the lack of large collections of this type of structured documents limits the extent of our research. Before we start explaining problems of information retrieval in such structured documents, a few remarks about the terminology for describing structured documents and structured retrieval are given.

A structured *collection* can be defined as a set of structured documents (files). At the lowest level of granularity in structured documents, we can specify the *content* of a structured document which represents all the words in a document that are not markup. On a higher level we can define structure *elements* or *components* that correspond to one of the *tags* or *markups* and all the encompassed information in it (including other contained tags and their content information). In contrast to *document retrieval* in flat text IR, for structured IR the emphasis is on *element retrieval*, i.e., *document component retrieval.* This difference is an important issue that is addressed in this thesis.

Identifying structured IR problems and opportunities

Structured queries are usually expressed using the extensions of structured query languages, such as XQuery [18] and XPath [14, 37] (XPath is an integral part of XQuery) for XML data. The aim of the extensions is to express IR-like search specifications over structured documents. Such extensions are the *about* clause in the Narrowed Extended XPath I (NEXI) query language [209], equality expression (=) in XIRQL [71, 72], or *ft:score* in the XQuery full-text extension [7].

For example, the simple information need where the user searches the XML collection for an `image` or a `video` depicting a 'garden' with 'flowers' in a `story` that contains `paragraph`s talking about 'beautiful flowers' and 'children' (see Figures 1.1 and 1.2), can be expressed in the NEXI query language as:

```
//story[about(.//paragraph, ‘‘beautiful flowers’’ children)]
                   //(image|video)[about(., flowers garden)]
```

Although this is a relatively simple example, several additional query capabilities that are not recognized in flat text retrieval can be illustrated with it:

1. The search for information can be performed in arbitrary parts of the structured document (or collection of structured documents) denoted with tags.

These tags are called *search elements* (or support elements). In our example, the search elements are paragraphs, images, and videos.

2. The *answer element* (or target element) that the user wants to see as an answer to his query can also be an arbitrary part of the document. Unlike in flat text retrieval where the search element (document) is the answer element, this is often not the case in structured retrieval. In our example query, image and video elements are the answer (as well as search) elements at the same time (denoted with '.' in the second *about* clause in our example query).

3. The user can perform searches in different XML elements and later combine their relevance scores for the answer element. In our example query the search is performed in paragraphs in a story element using the terms 'beautiful flowers' and 'children' and then combined with the search for the terms 'flowers' and 'garden' in image or video elements (denoted with the '|' symbol in the example NEXI query).

Additionally, looking at the document representation, simple inverted indexes (lists of term-document pairs) [90] are not sufficient to store all the information present in structured documents. That is why for structured information retrieval a number of new indexing methods has been proposed, ranging from modified inverted index structures [79, 128, 155] to full database implementations [66, 77, 164], each having its advantages and pitfalls. The complexity of the physical representation of documents in structured IR is an important reason why the areas of *information retrieval* and *databases* (*DB*) are brought closer to each other (see also Section 1.2.2).

The central part of any retrieval system, and therefore a structured IR system as well, is the retrieval model that performs the matching (comparison). However, for structured IR, the retrieval model has to incorporate additional query capabilities that are present in the query representation, as well as the new complex indexing structures used for structured document representation. This is why structured information retrieval models introduce reasoning over structure, i.e., over search and answer elements, in addition to reasoning over terms. To cope with such complex retrieval problems, many structured retrieval approaches upgrade the best performing flat text retrieval models, such as tf.idf [193], BM25 [181], or language models [95], with structured reasoning [89, 108, 155], or new retrieval models specifically aimed for structured retrieval are developed, e.g., [45, 198]. The ultimate goal of both classes of approaches is to make a model that would enable effective (and efficient) structured information retrieval.

Out of many properties of traditional flat text IR systems, we emphasize the following four that are highly relevant to structured IR:

1. Traditional IR systems are developed for a simple query language, consisting in most cases of a list of terms, and are not suitable for complex queries such as our example query.

2. Traditional IR systems are retrieval model specific, i.e., the retrieval model (or a small class of similar retrieval models) is hard-coded in the system, and its adaptation to structured IR is not straightforward.

3. Traditional IR systems are developed for a specific storage structure, in most cases inverted indexes, and cannot be easily adjusted to support storage structures needed for structured IR.

4. Traditional IR systems lack the notion of *data independence* [69]: any change in the document storage structures or any change in the retrieval model used, will lead to system developers needing to change major parts of the system.

Following these properties, the implementation of specialized IR systems is faster and simpler than any implementation of a *database management system* (DBMS) on ranked retrieval tasks. However, these properties become drawbacks when extending flat text retrieval systems to support structured IR. How databases can handle such structured IR problems is explained below.

1.2.2 DB systems and structured retrieval

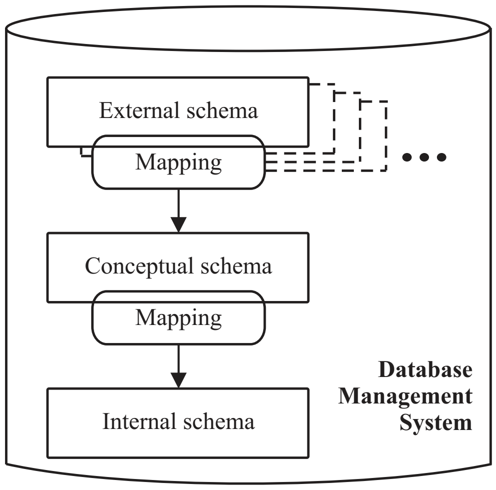
Before we explain structured information retrieval from a database point of view, we introduce the terminology that is used throughout the thesis. A *data model* is a model that describes in an abstract way how data from a conceptual model of some real-world scenario is represented in, e.g., a database. Different data models can be used to define the same real-world area. For example, in the database area different models are used: relational model, object-oriented model, hierarchical model, and network model. They exist because different real-world areas are better described by different data models.

The data model is used as a base for defining an algebra. Algebra is a branch of mathematics that is composed of a *domain of values* and a set of *operators* that operate on those values. For example, the domain of Boolean algebra are truth values: TRUE or FALSE, and the operators are: AND, OR, and NOT. The most important property of an algebra is that it is closed in the domain, i.e., the result of the application of algebra operators is always in the domain of values. The operators frequently follow some *properties*, such as associativity or commutativity.

DBMS architecture

Most database systems use well established database operators and thirty years of experience in relational database management systems (RDBMS). The main characteristic of the database approach is its layered architecture with a strong separation between *external*, *conceptual*, and *internal* schema, as described in the ANSI/X3/SPARC model [210] depicted in Figure 1.4.

Figure 1.4: The three-schema DBMS architecture.



Each schema might be built upon its own data model and different algebraic operators or applications might be used at different levels. External schemas are application specific and application programs are defined to support different data views. The conceptual schema specifies a high-level view on 'modeling the world' (e.g., relational model in a RDBMS). The implementation of queries over such a schema is usually based on an algebra, such as using relational algebra for RDBMSs. The internal schema conveys all the details of the data storage and handling at the physical medium.

By using different data models for each schema, *data independence* is provided. The data independence consists of:

- *logical data independence*: the possibility to change the conceptual schema without having to change the external schemas, i.e., user application programs;
- *physical data independence*: the possibility to change the internal schema, i.e., storage and access structures, without having to change the conceptual schema.

The central part that provides the data independence is the conceptual schema. When using different data models for external and internal schemas, the only part of the system that needs to be changed is the interface that performs the *mapping* between the schemas. For example, if different storage structures are used, only the mapping module that communicates between the internal and conceptual schemas needs to be changed. This results in a great *flexibility* that is the major property of the DBMS architecture.

A database approach to structured IR

Comparing Figures 1.3 and 1.4, we can conclude that while for IR systems the matching is an inseparable concept or action performed on relatively simple query and document representations, 'DB matching' has to be done in several steps, using different representations at each database level. Although the database approach seems more complex than simple flat text matching, it gives opportunities for the more powerful matching specification in structured IR.

For (structured) IR systems, following this separation in levels, would give another, additional advantage over flat text IR systems: by choosing the appropriate level of abstraction for each database schema, the development of scoring techniques, handling structured information, would be kept *transparent* for the rest of the system design. Furthermore, it would make it flexible with respect to the query language, retrieval model, and physical implementation. In other words *content independence* [55, 59], the independence of the development of the end-user programs that access the content using the external schema from the actual document representation using the internal schema, is supported.

Similarly to logical and physical data independence, in this thesis we address two new forms of content independence, that emphasize the importance of the database approach to structured IR:

- *retrieval model independence*: the possibility to use different retrieval models when implementing the conceptual schema in a database, without having to change external applications and query languages used;

- *content description independence*: the possibility to use different internal data (document) representations without having to change the conceptual schema and the specification of algebra operators that define retrieval models in conceptual schema.

To abstract away from the structured IR implementations within external and internal schemas, we introduce a mathematical framework (algebra) that implements the conceptual schema. The conceptual schema and (logical) algebra can be identified as the central component of an integrated DBMS and IR system for providing content independence, similar to the role of relational algebra in enabling data independence in RDBMSs. Additionally, such an algebra could use the reasoning over algebra operators for query rewriting and *optimization*, as in [16].

1.2.3 Vision of the future retrieval systems

Judging by today's distribution of different types of documents (differently structured documents) on the web, the future information sources will be centered around structured textual documents (HTML, XML, SGML, PDF, MPEG-7, MPEG-21, etc.), images (jpeg, png, gif, etc.), audio material (mp3, wav, etc.), and video material (avi, MPEG-4, divx, etc.). Therefore, *modern information retrieval* systems need to provide effective retrieval for the mixture of various types of documents, consisting of textual documents, images, and audio-video material.

To identify the elementary features that need to be supported in modern information retrieval systems, the simple example query from Section 1.1 is modified in the following way:

```
//story[about(.//paragraph, "beautiful flowers" children) or
             about(.//video, garden src:garden.avi)]
    //image[about(., flowers garden) and
             about(., src:flower_garden.jpg)]
```

Now the user searches for a story containing either paragraphs about 'beautiful flowers' and 'children', or video clips about 'garden' and ones that are similar to a video clip that he/she liked, stored as 'garden.avi'. He/she would like to get as answers images in such stories that are about 'flowers' and 'garden' and that look like an image that he/she found nice, named 'flower_garden.jpg' (e.g., similar to an image in Figure 1.2).

This query illustrates four features that need to be supported in modern IR systems: entity selection, score computation, score combination, and score propagation.

First, as can be seen from the example query, a retrieval system needs to model different types of document content. These are entities, such as elements in a document structure: stories, paragraphs, videos, images, terms: 'beautiful flowers' and 'children', and multimedia content: 'garden.avi' and 'flower_garden.jpg'. Looking from the retrieval perspective, these entities need to be *selected*. While modeling (or selecting) terms and markup is easy, modeling (or selecting) multimedia data is, by far, more complex. Furthermore, the modeling does not have to be unique and can change over time. This is why the database approach, having content description independence is beneficial.

Second, relevance scores of certain document components need to be determined with respect to the entities or multimedia content they contain: terms, elements, images, videos. In our example, relevance of a paragraph element should be determined based on the terms it contains. It is relevant if it contains the term 'beautiful flowers' and/or 'children'. Similarly, a video element is relevant if it is similar to a source video 'garden.avi' and an image element is relevant if it contains a picture similar to 'flower_garden.jpg'. Due to different entity representations the element *relevance score computation* can be quite diverse. This is why retrieval model independence is an important issue in modern IR systems.

Third, the relevance scores obtained for different document entities need to be *combined*. For example, scores for paragraphs about 'beautiful flowers' or 'children' should be combined with scores of a video similar to 'garden.avi'. This implies that the system should be flexible with respect to a combination of different document component relevance score computations.

Fourth, retrieval systems should also enable the *propagation* of scores between different entities, such as from video to story elements or from image to story elements, as shown in the example query.

To be effective on retrieval from heterogeneous collections and to support different user requests, an IR system needs to be flexible with respect to the specification of different retrieval models and various document and query representations. It should also support *elementary requirements for structured retrieval* that are discussed above, i.e., selection of different entities (entity selection) as well as computation, combination, and propagation of relevance scores.

1.3 Research questions and research methods

The research presented in this thesis is focused on developing a flexible structured IR framework, with a logical algebra (implementing the middle layer of Figure 1.4) as its central part. The algebra (framework) should support transparent instantiation of retrieval models by supporting retrieval model independence and content description independence. The framework is used as an experimental platform for developing effective structured retrieval models applicable to different domains, such as text documents, images, and videos.

The main goal of the research presented in this thesis is to give answers to the following three questions.

Q1 To what extent can a logical algebra for structured IR support retrieval model independence?

Due to the advantages of the database approach discussed in Section 1.2.2, we follow a layered architecture in designing our structured IR system. As opposed to the flat text IR implementations that do not recognize the need for layered architecture, in the database approach the conceptual schema can be considered as the central part that should provide the flexibility in the structured information retrieval framework. The conceptual schema is described in terms of a formal algebra, and the algebra we develop is called *score region algebra*. The overall goal of the algebra is to enable *transparent* specification of retrieval models, i.e., to abstract away from the specific retrieval model used. The algebra assumes that the scoring mechanism (ranking) is part of the algebra and not a side effect of performing some algebraic operations or employing a separate IR module.

The algebra is developed following the guidelines for the development of modern information retrieval systems described in Section 1.2.3. It supports different query formulations and search and answer element specification, and follows the four elementary structured retrieval requirements: entity selection, score computation, score combination and score propagation. Additionally, it does not make any restriction on the definition of retrieval models, physical implementations, and query languages used. It supports content and data independence, but most importantly *retrieval model independence*.

Q2 What is the influence of different retrieval model instantiations within the algebra on the effectiveness of: a) document component retrieval and b) document retrieval?

a) Structured queries greatly vary in their complexity: ranging from simple list of term IR queries or field search queries to queries that heavily exploit document structure. The main goal of a structured IR system is to be effective on retrieval tasks using such diverse queries. To test the usefulness of our logical algebra, different retrieval models are instantiated in the algebra and tested on effectiveness. For the elementary structured IR analysis, complex structured queries are used that specify a combination of searches in different parts of structured documents as well as specification of searches in nested structures. The ultimate goal is not to come up with the best retrieval model for structured IR but with the guidelines that can help researchers developing their own retrieval systems or when they use our logical algebra to instantiate new, better retrieval models following the four elementary retrieval requirements.

b) To test the effectiveness of our framework for document retrieval a subclass of structured queries is evaluated and compared to unstructured queries. These structured queries are represented as a Boolean combination of terms (Boolean queries), and as simple structured queries that use specific tags inside documents, i.e., fields, to express more focused search (field search queries). The flexibility of the logical algebra for modeling document retrieval using unstructured and structured queries, as well as different retrieval models, is demonstrated on shallowly structured (poorly and not deeply nested) documents. The goal is to detect retrieval models that are effective in such retrieval tasks in shallowly structured documents, and to find what is the best way of incorporating additional structured information in traditional document retrieval.

Q3 Can the algebra be extended to: a) support ranked retrieval using richer data models and b) provide effective retrieval in domains other than text?

a) The algebra needs to be defined in such a way that it can easily be extended to support richer data models. It should be possible to extend the data model with additional information items extracted from structured documents that would not override the specification of algebraic operators and their properties. The added information items should be used for the specification of new retrieval models following the structured retrieval requirements, or for extending the operator set. For illustration we present the extensions of the algebra data model and operator set for incorporating the nesting level of elements. The effectiveness of retrieval models that use these extensions is also discussed.

b) The algebra should be developed in such a way that it can incorporate different types of information (features) extracted from non-textual documents. In other words, it should be possible to integrate information from heterogeneous data sources while keeping the same algebraic framework. As an example, the integration of video and image content in the algebraic framework is explained. Video and image search is integrated with text search in the algebra following the data independence and content independence criteria. Additionally, the effectiveness of such multimedia retrieval within the algebraic framework is evaluated.

1.4 Outline

This thesis is organized as follows.

Chapter 2 discusses related work and approaches that are used as an inspiration for our research. It gives a brief introduction to various information retrieval models and presents the adaptations of these models to structured information retrieval. It then tries to close the gap between information retrieval and database research fields, having structured document (XML) retrieval as its scope. After that, a number of region algebra approaches are presented and their features and characteristics are discussed, as they form the base for the logical algebra defined in this thesis.

Chapter 3 explains in detail the four elementary requirements for developing a structured information retrieval system. It starts with illustrating the need for the identification of these requirements and discusses what they model in structured IR. Each requirement, namely entity selection, score computation, score combination, and score propagation is discussed in isolation. The chapter then introduces *score region algebra*, a logical algebra that is the central part of our retrieval system. Score region algebra operators are defined following the four elementary retrieval requirements and their functionality is explained. After that, the chapter gives justification for the specific type of algebraic operators that are used in score region algebra, and presents alternative approaches. The chapter is concluded with a discussion on opportunities and limitations of score region algebra framework on structured retrieval tasks.

Transparent instantiation of retrieval models is the main topic of Chapter 4. It first presents our three-level database system called *TIJAH*, and explains its layered architecture. The chapter continues with the formal description of the retrieval models that are instantiated within the algebraic operators, following the four elementary retrieval requirements. The properties of algebraic operators having different retrieval model implementations are explained here. They illustrate the opportunities for logical query optimization using score region algebra operator properties.

Chapter 5 covers a large set of experiments on highly structured documents (XML). It starts with presenting the aim of the experiments discussed in the chapter and with describing the test collection and experimental setup. First, the results of experiments on the elements' size, length prior, and parameters of different retrieval models are discussed. Then, each retrieval aspect is analyzed on a specific set of topics, ranging from simple ones where score combination and score computation are examined to complex ones where downwards and upwards score propagation is investigated. The chapter ends with a discussion on experimental results and recommendations on what future retrieval systems should support when modeling structured retrieval, following the four elementary retrieval requirements.

Chapter 6 presents a set of experiments on shallowly structured documents. The aim is to determine whether structure in queries and in documents helps, and if it does in what way. A short description of the test collection and experimental setup is given after presenting the aim of the experiments discussed in the chapter. The influence of computation of elements' size, length prior, and parameters of different retrieval models are then presented. The chapter continues with the comparison between the effectiveness of unstructured and structured queries. It then shows that the usage of structured queries, even on shallowly structured documents, improves effectiveness. At the end, results are discussed and the usefulness of structured queries and structured documents are pointed out.

Chapter 7 shows how score region algebra can be extended to support richer data models and different data domains. It starts with the usage of the element nesting level information for specifying retrieval models. Video search modeled as a search on structured speech transcripts is presented afterward. The chapter then discusses the extensions of the algebra introduced to support image search and presents the effectiveness of a combined image and text search scenario. The chapter is concluded with a short overview of the presented extensions and with summarizing the effectiveness of introduced retrieval models.

Finally, Chapter 8 concludes this thesis. It follows the three research questions introduced in the previous section. The chapter emphasizes what are the benefits of using the flexible algebraic framework, i.e., being able to easily instantiate different retrieval models and to easily extend the algebra to other domains. It also discusses the outcome of the numerous experiments performed using our transparent logical algebra. The directions for further research are also presented in this chapter, with respect to each research question.

Chapter 2

Document Component Retrieval

Until the last decade of the 20^{th} century the information on document structure and semantic annotation of documents was not fully exploited in the retrieval process. The early work by Callan, Turtle, and Croft [29, 50] on incorporating hypertext links and field-based queries into retrieval systems, and Salminen and Tompa [188] and Burkowski and Clarke [27, 38] on structured text search, showed its usefulness and set the path for future research. Furthermore, it brought together researchers from information retrieval as well as database field in resolving the issues of effective and efficient information retrieval in structured documents, especially XML (see [125] for an overview). On the other hand, several initiatives are started for studying information retrieval in structured documents, such as INEX [74] and TREC Enterprise Track [220].

This chapter discusses the evolution of retrieval models from flat text retrieval models used for document retrieval to structured retrieval models used for document component retrieval. It then explains the database view on structured information retrieval, with a focus on XML, and presents the overview of approaches for structured retrieval, termed region algebras. The chapter is concluded with a short summary.

2.1 Information retrieval models

This section covers flat text retrieval models, i.e., retrieval models that disregard explicit or implicit document structure. It then presents some of the most important adaptations of the flat text models to structured retrieval. Additionally, a short overview of multimedia retrieval is given.

2.1.1 From Boolean to language models

The term information retrieval often implies ranked retrieval, as the main goal of an information retrieval system is to find documents *relevant* to the user information need and to *rank* them in order of predicted relevance to the user. The essential component of the retrieval system is a model that specifies the matching process, initiated by a query over a set of documents, and terminated by presenting the ranked list of relevant documents to the user. This model is often referred to as *retrieval model*. Following the IR process (depicted in Figure 1.3), the retrieval model must define a *relevance ranking function* based on a query representation and document representation.

A relevance ranking function in flat text retrieval systems is based on term statistics, used to estimate how relevant documents are to a user query. For example, the number of occurrences of a query term in the document can be used to estimate the relevance of a document. The greater the number of occurrences of a query term, i.e., the higher the *term frequency*, the more relevant a document. Additionally, term distributions, such as Zipf's distribution of terms in the collection [213], are used to improve the results of relevance ranking functions. The knowledge of distribution of terms can be implemented in the form of: (1) *stop words*, i.e., disregarding the most common words in the language such as 'the', 'a', 'as', or 'for' in English, (2) *document frequency*, i.e., number of documents in the collection that contain a query term, or (3) *collection frequency*, i.e., number of terms in the whole collection. Except for the Boolean model, these elementary term distribution statistics form the base of information retrieval models.

In the following we give a description of the several classes of retrieval models, namely Boolean, fuzzy set, vector space, probabilistic, inference network, and language models. Note that this is not the ultimate classification as other models exist, e.g., latent semantic indexing, neural networks, genetic algorithms (see [83]). As they are not exploited in this thesis, we do not focus on them.

Boolean models

The Boolean model was a predominant model in commercial information retrieval systems until the 1990's. It is based on set theory and Boolean algebra. Documents are represented as a set of terms and queries as a Boolean expression on terms. The term can be either present or not in a document, and its presence/absence determines whether that document is retrieved or not. Queries are expressed using query terms and operators: AND, OR, and NOT. For example, for the Boolean query:

```
((beautiful AND flowers) OR bouquet) AND NOT(vase)
```

only documents that contain terms 'beautiful' and 'flowers', or the term 'bouquet', and do not contain the term 'vase', are presented to the user.

Although the model is simple and based on a widely used and easy to understand (and implement) set theory, the Boolean model has many drawbacks. The Boolean formalism only supports the exact matching and therefore can result in an empty result set or overloaded output [43]. In Boolean retrieval retrieved documents are not ranked. Furthermore, all terms in the query as well as in the document are considered equally important.

To overcome some deficiencies of the Boolean model several extensions have been proposed. Salton, Fox, and Wu [192] used the idea that documents for the query AND expression should be scored according to their distance (dst) from the ideal (document or query) point that has the value 1, while documents for the query OR expression should be scored according to their distance from the ideal point with value 0. Generalizing the idea, the authors developed the *p-norm* model depicted in Equation 2.1, where $S(doc, q)$ is a relevance score of a document doc; $tm_i,\ i = 1, ..., n$, denotes query terms in a query q, and p is a parameter that needs to be empirically estimated.

$$S(doc, tm_1 \text{ AND } tm_2 \text{ AND ... AND } tm_n) = 1 - \left(\frac{\sum_{i=1}^{n} (dst_{tm_i})^p \cdot (1 - dst_{doc_i})^p}{\sum_{i=1}^{n} (dst_{tm_i})^p} \right)^{1/p}$$

$$S(doc, tm_1 \text{ OR } tm_2 \text{ OR ... OR } tm_n) = \left(\frac{\sum_{i=1}^{n} (dst_{tm_i})^p \cdot (dst_{doc_i})^p}{\sum_{i=1}^{n} (dst_{tm_i})^p} \right)^{1/p} \quad (2.1)$$

Although overcoming deficiencies of the Boolean model, the p-norm approach does not provide a framework for determining term and document distances, i.e., dst_{tm_i} and dst_{doc_i}.

Fuzzy set models

Unlike in the Boolean model where documents can be either in a relevant set or not, fuzzy set models [119] allow the definition of the *degree of membership*, denoted with T, for representing the relevance of a document to a query term. The rules for the membership degree of an expression, consisting of Boolean combination of query terms, are based on 'fuzzy' conjunction, disjunction, and negation operators, given below.

$$\begin{aligned} T(tm_1 \text{ AND } tm_2 \text{ AND ... AND } tm_n) &= min(T(tm_1), T(tm_2), ..., T(tm_n)) \\ T(tm_1 \text{ OR } tm_2 \text{ OR ... OR } tm_n) &= max(T(tm_1), T(tm_2), ..., T(tm_n)) \\ T(\text{NOT } tm) &= 1 - T(tm) \qquad (2.2) \end{aligned}$$

Although simplistic, the fuzzy set model does not justify why such definitions of membership functions are used. Furthermore, it is not sensitive to the real distribution of query terms, especially for long queries. For example, in the query expression: tm_1 OR tm_2 OR ... OR tm_n, where $T(tm_1) > T(tm_j),\ j = 2, ..., n$, document relevance score would be the same irrespective of what are the degrees of membership for terms other than tm_1. The values of $T(tm_j)$ might be just as high as $T(tm_1)$ or they can all be zero. To overcome this deficiency, Paice [160] modified fuzzy OR and AND, and used the formula given in Equation 2.3. In the formula the summation is performed in the decreasing order of the degree of membership for OR and in the increasing order for AND queries. The parameter k is the degree of 'softness' of the operator and has to be estimated empirically.

$$S(doc, q) = \frac{\sum_{i=1}^{n} k^{i-1} T(tm_i)}{\sum_{i=1}^{n} k^{i-1}} \tag{2.3}$$

The advantage of fuzzy set models over the Boolean model is that they allow the computation of relevance scores for a document, and therefore support relevance ranking. However, they rely on a term weighting algorithm for estimating the degree of membership but do not give any hints or justification how the degree of membership should be derived.

Vector space models

In the vector space model [193] queries and terms use a 'bag-of-words' representation, where documents (d) and queries (q) are represented as *vectors* of term weights (w): $d = \{w_{d,1}, w_{d,2}, ..., w_{d,m}\}$ and $q = \{w_{q,1}, w_{q,2}, ..., w_{q,n}\}$. The relevance estimation is based on a similarity between a document vector and a query vector, measured by a cosine of the angle between them, as shown in Equation 2.4. Documents are then ranked in decreasing value of the similarity measure.

$$sim(d, q) \ = \cos \sphericalangle(d, q) = \frac{\overrightarrow{d} \cdot \overrightarrow{q}}{|d| \cdot |q|} = \frac{\sum_j w_{d,j} \cdot w_{q,j}}{\sqrt{\sum_j w_{d,j}^2} \cdot \sqrt{\sum_j w_{q,j}^2}} \tag{2.4}$$

Equation 2.4 shows that the higher the weight of the term the greater the impact on the relevance score. The open question is how to assign the weight to each term. Weights are specified following the two design goals: (1) reward terms that best describe the document, and (2) punish terms that do not distinguish among documents in the collection. In flat text IR this corresponds to *term frequency* and *inverse document frequency* respectively. The term frequency (*tf*) is estimated as a number of occurrences of a term in a document, while the inverse document frequency (*idf*) is usually defined as a natural logarithm of the inverse of the number of documents that contain the term.

tf.idf Assuming that the weights of query and document terms are estimated by the term frequency and the inverse document frequency, we can define the similarity between the query q, consisting of query terms $tm_i, i = 1, 2, ..., n$, and the document doc as shown in Equation 2.5. Here $tc(tm_i, doc)$ denotes the number of occurrences of a term tm_i in the document doc (i.e., term count), while $dc(tm_i)$ denotes the number of documents containing the term tm_i (i.e., document count).

The number of documents in the collection is denoted with N. This is the base for most vector space models used in information retrieval.

$$S(doc, q) = \sum_{i=1}^{n} tc(tm_i, doc) \cdot ln \frac{N}{dc(tm_i)} \tag{2.5}$$

However, many variations of the basic formula are derived later [24, 185, 190]. For example, the *tf* part can be computed as:

$$\frac{tc(tm_i, doc)}{max_j(tc(tm_j, doc))},\ 0.5 + \frac{0.5 \cdot tc(tm_i, doc)}{max_j(tc(tm_j, doc))},\ \text{or } 1 + ln(tc(tm_i, doc)),$$

while *idf* can be computed as:

$$ln\frac{N+1}{dc(tm_i)},\ ln\left(1 + \frac{N}{dc(tm_i)}\right),\ \text{or } ln\left(\frac{N - dc(tm_i)}{dc(tm_i)}\right).$$

The basic vector space model and its variants enable ranked retrieval, allow for the term weighting, and support partial matching. However, the weighting in the model is intuitive and it is not based on a clear mathematical formalism.

Probabilistic models

In the probabilistic model, pioneered by Maron and Kuhns [126], IR is defined as an uncertain process of inferring the *probability of relevance* based on documents and queries [67, 70]. It ranks documents in decreasing order of probability of relevance $\mathcal{R}$ to the information need, denoted as $P(\mathcal{R}|q, doc)$. The probability of relevance is estimated using the *retrieval status value* function $rsv(doc, q)$ that depicts the ratio of probability of relevance of a document $P(\mathcal{R}|doc)$ to a probability of non-relevance of a document $P(\overline{\mathcal{R}}|doc)$. Using Bayes' rule [161] and the binary independence assumption (i.e., terms occur independently of each other given that we know their relevance $\mathcal{R}$ and non-relevance $\overline{\mathcal{R}}$), and removing the constant values $P(\mathcal{R})$ and $P(\overline{\mathcal{R}})$, the probabilistic model can be described by Equation 2.6.

$$rsv(doc, q) = \sum_{i=1}^{n} ln \frac{P(tm_i|\mathcal{R}) \cdot P(\overline{tm_i}|\overline{\mathcal{R}})}{P(tm_i|\overline{\mathcal{R}}) \cdot P(\overline{tm_i}|\mathcal{R})} \tag{2.6}$$

In Equation 2.6, $P(tm_i|\mathcal{R})$ is the number of relevant documents that contain a query term tm_i, divided by a number of relevant documents: $|doc_{tm_i} \cap doc_{rel}|/|doc_{rel}|$; $P(\overline{tm_i}|\overline{\mathcal{R}})$ is the number of non-relevant documents that also do not contain the term, divided by the number of non-relevant documents: $(N - |doc_{tm_i} \cup doc_{rel}|)/(N - |doc_{rel}|)$; $P(tm_i|\overline{\mathcal{R}})$ is the number of documents that do contain the query term but are not relevant, over the number of non-relevant documents: $|doc_{tm_i} \backslash doc_{rel}|/(N - |doc_{rel}|)$; $P(\overline{tm_i}|\mathcal{R})$ is the number of relevant documents that do not contain the term, divided by the number of relevant documents: $|doc_{rel} \backslash doc_{tm_i}|/|doc_{rel}|$. Equation 2.6 can be simplified to an *idf*-like formula given in Equation 2.7 (see page 21 of [95]). In the formula, the following notation is used: $rr_i = |doc_{tm_i} \cap doc_{rel}|$, $dc(tm_i) = |doc_{tm_i}|$, and $rel = |doc_{rel}|$.

$$rsv(doc, q) = \sum_{i=1}^{n} ln \frac{(rr_i + 0.5) \cdot (N - rel - dc(tm_i, doc) + rr_i + 0.5)}{(dc(tm_i, doc) - rr_i + 0.5) \cdot (rel - rr_i + 0.5)} \tag{2.7}$$

The main drawback of this formula is that it does not incorporate the *tf* part which results in poor retrieval performance [181].

BM25 The best match (BM) models are based on the probabilistic formula given in Equation 2.7. Robertson and Walker [181] tried to overcome the absence of the *tf* part in the formula by using a mixture of two Poisson distributions for relevant and irrelevant documents. They tried many variations of the formula among which the *BM25* showed the best performance. Later, this model is used in the Okapi [181], as well as in the Inquery [30] retrieval models. The basic formula is given in Equation 2.8, where *avdl* is the average document length, and k_1 (between 1.0 and 2.0), k_3 (between 0 and 1000), and b (between 0.6 and 0.75) are constants.

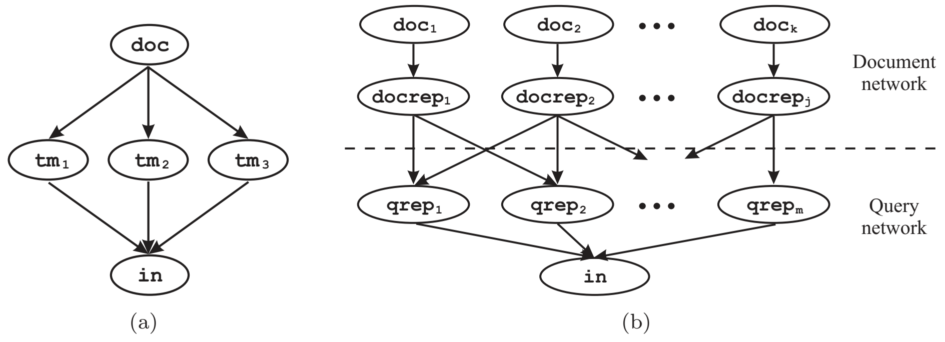
$$\begin{aligned} S(doc, q) = & \sum_{i=1}^{n} (ln \frac{N - dc(tm_i) + 0.5}{dc(tm_i) + 0.5} \cdot \\ & \cdot \frac{(k_1 + 1) \cdot tc(tm_i, doc)}{k_1((1 - b) + b\frac{len(doc)}{avdl}) + tc(tm_i, doc)} \cdot \frac{(k_3 + 1) \cdot tc(tm_i, q)}{k_3 + tc(tm_i, q)}) \end{aligned} \tag{2.8}$$

The features of the BM25 model are that it: (1) unifies probabilistic and *tf.idf* weighting in a mathematical framework, (2) uses separate document and query length normalization, and (3) most importantly, introduces several tuning constants which can be trained on different collections. Note also that in the Inquery system, several functions are implemented to combine the scores of a single term relevance score computations, such as sum, multiplication, or probabilistic sum (see below).

Inference network models

Inference network models [29, 30, 50, 211] are based on *Bayesian networks*. A Bayesian network [161] is an *acyclic directed graph* that encodes probabilistic dependency relationships among random variables. A simple Bayesian network for

Figure 2.1: Inference networks for information retrieval: (a) A simple Bayesian network for document retrieval; (b) Inquery inference network.



IR is represented in Figure 2.2(a). It represents the conditional dependence between a document *doc*, query terms $\{tm_1, tm_2, tm_3\}$, and information need node *in*. In the figure, terms are independent given their parent node, i.e., document.

All nodes in the network represent binary random variables with values $\{0, 1\}$, while all edges represent probabilities depicting conditional dependencies among nodes, usually specified in a table called *conditional probability table*. The computation of the joint probability for all nodes in the graph can be simplified using the conditional independence assumption, as presented in Equation 2.9.

$$\begin{aligned} P(doc, tm_1, tm_2, tm_3, in) = & \\ & P(doc)P(tm_1|doc)P(tm_2|doc)P(tm_3|doc)P(in|tm_1, tm_2, tm_3) \end{aligned} \tag{2.9}$$

Documents should be ranked according to the probability of information need fulfillment, assuming that the document is relevant: $P(in = 1, doc = 1)$. In general, one needs to specify the conditional probability tables for $P(tm_i|doc)$, $i = 1, ..., n$, and $P(in|tm_1, tm_2, ..., tm_n)$. The major problem in the model is that the conditional dependence tables have exponential growth with the number of query terms (n), and that the model assumes an ad hoc estimation of conditional probabilities.

Inquery and Indri To cope with the problem of intractable inference Callan and Croft proposed a retrieval model based on a version of Bayesian network that consists of two layers, document network and query network, as depicted in Figure 2.2(b). This retrieval model is used in the Inquery retrieval system [23, 29, 30]. The document network specifies document representation nodes (*docrep*) and their conditional dependence on document nodes (*doc*). The query network consists of query representation nodes ($qrep$), and information need node (in) that represents the event the information need is met. The query network, i.e., query representation nodes, is connected with the document representation nodes by numerous edges.

In the original model it is assumed that document and information need node are considered to have a value 1 or *true*. Furthermore, authors simplified the conditional dependence network between document and query representation nodes, allowing only canonical forms of $P(qrep|docrep_1, docrep_2, ..., docrep_n)$ for AND, OR, NOT, SUM, MAX, and WSUM (weighted sum), depicted in Equation 2.10. The parameters $wg_i,\ i = 1, ..., n$, represent query term weighting factors. The conditional probabilities $P(docrep_i|doc_i),\ i = 1, ..., n$, are estimated based on the Okapi model [182].

$$\begin{aligned}
P_{AND}(in = 1|doc = 1) &= p_1 \cdot p_2 \cdot ... \cdot p_n \\
P_{OR}(in = 1|doc = 1) &= 1 - (1 - p_1) \cdot (1 - p_2) \cdot ... \cdot (1 - p_n) \\
P_{NOT}(in = 1|doc = 1) &= 1 - p \\
P_{MAX}(in = 1|doc = 1) &= max(p_1, p_2, ..., p_n) \\
P_{SUM}(in = 1|doc = 1) &= \frac{p_1 + p_2 + ... + p_n}{n} \\
P_{WSUM}(in = 1|doc = 1) &= \frac{wg_1 \cdot p_1 + wg_2 \cdot p_2 + ... + wg_n \cdot p_n}{wg_1 + wg_2 + ... + wg_n}
\end{aligned} \tag{2.10}$$

To add more formal justification for the estimation of conditional probabilities, inference network model is fused with the language model [134] which resulted in the new retrieval system called Indri [207]. This model uses language modeling for estimating the *docrep* probabilities, and it extends the basic canonical set of belief operators with operators such as MIN and $WAND$ (weighted AND).

Statistical language models

Statistical language models for information retrieval are an adaptation of a hidden Markov model [107, 146]. A statistical language model (LM) is used to estimate the probability distribution $P(str)$ over strings. It attempts to reflect how frequently a string *str* occurs. This approach is widely used for speech recognition [107].

For information retrieval two classes of language models can be distinguished: generative language models and divergence language models [154]. The divergence language model is based on Kullback-Leibler divergence [153] and it measures how much a query language model diverges from the document language model. The generative language model is based on estimating the probability that each document generated the query string [95, 171]. As the latter one is more frequently used in IR approaches we describe it here.

The most frequently used generative LM is the *n-gram* language model. It expresses the probability of generating a string *str* consisting of terms $tm_i \in$

str, $i = 1, .., m$, as shown in the Equation 2.11, assuming that the probability of generating a word depends on a $n-1$ preceding words: $P(tm_k|tm_{k-n+1}, ..., tm_{k-1})$, where n is usually greater than 2.

$$P(str) = P(tm_1)P(tm_2|tm_1)...P(tm_m|, tm_{m-n+1}, tm_{m-n+2}, ..., tm_{m-1}) \quad (2.11)$$

For the efficient evaluation, the n-gram model is simplified to a *bigram* and *unigram* models. While the bigram model assumes only the dependence of a term on a word preceding it, the unigram model assumes independence between terms in a document. The unigram language model is among the most used models in todays information retrieval systems [115, 117, 135, 205, 206] and it is also used for passage retrieval [124], web page retrieval [109], etc. Its specification is given in Equation 2.12.

$$P(str|doc) = P(tm_1|doc)P(tm_2|doc)...P(tm_n|doc) \quad (2.12)$$

The language model proposed by Hiemstra [95] starts with the question how relevant is a document *doc* given a query *q*. Using the unigram language model paradigm, Bayes' rule, and the fact that the probability of generating the query $P(q)$ is uniformly distributed for all queries, he derives the language model depicted in Equation 2.13.

$$P(doc|q) = \frac{P(q|doc) \cdot P(doc)}{P(q)} \approx \prod_{i=1}^{n} (P(tm_i|doc)) \cdot P(doc) \quad (2.13)$$

The probability of generating the query term from a document $P(tm_i|doc)$ is computed using the maximum likelihood estimate, i.e., the term frequency divided by document length. Additionally, to incorporate the term distribution factor in ranked retrieval and to avoid the sparse data problem [95], the probability also includes the re-estimation, i.e., smoothing, based on one of the smoothing methods [223]. Using, e.g., the linear interpolation (Jellinek-Mercer) smoothing, background statistics are added to the model. The background statistics are based on a number of terms in the whole collection (*col*), i.e., *collection frequency*, or a number of documents containing the term, i.e., *document frequency*. In the former case the relevance score of the document (*doc*) given the query terms (tm_i, $i = 1, 2, ..., n$, $tm_i \in q$) can be computed as:

$$P(doc|q) \approx \prod_{i=1}^{n} \left(\lambda \frac{tc(tm_i, doc)}{len(doc)} + (1-\lambda) \frac{tc(tm_i, col)}{len(col)} \right) \cdot \frac{len(doc)}{len(col)} \quad (2.14)$$

where $len(doc)$ and $len(col)$ are the length of a document and a collection (i.e., the number of terms they contain) respectively, and λ is a smoothing parameter (ranging from 0 to 1) that specifies the relative influence of the foreground (term frequency) and background statistics in the final document relevance score computation.

The drawback of statistical (unigram) language models is that they use little knowledge of what the language really is [187], resulting in difficulty to improve the effectiveness of the model without the significant re-modelling. For improving the models, Rosenfeld et al. proposed the usage of dependency models [35, 118] or sentence models [186], and their usefulness is still studied.

2.1.2 Retrieval models for structured data

Information retrieval (IR) theory is developed to overcome the task of searching for information in unstructured (flat text) documents. The theory and the tools used in conventional IR systems usually completely disregard the structure of a document. However, with the rapid expansion of structured documents, especially XML, a new research topic has emerged. This topic is concerned with the formulation of more complex, structured queries, as well as with the search and retrieval of information conveyed in document components.

With the term *structured queries* we denote not only the queries that express relation among query terms such as Boolean and proximity queries, but also the the queries that include restrictions on where these terms should be located in the document, i.e., structured constraints. *Document components*, also called *elements*, correspond to segments of a document that can be used for information search or as units of retrieval. Document components can correspond to sentences, paragraphs, sections, or various elements in marked up documents.

The origin of document component retrieval can be found in the research area of passage retrieval (see [189] for an overview of the earliest papers and [105] for the more recent ones). The goal of passage retrieval is to identify and extract text fragments from full-text documents that are relevant to a user query. These text fragments are composed of coherent relevant sentences or paragraphs. Passage retrieval techniques are widely used for text summarization (e.g., [159]) and question answering (e.g., [208]). However, in this thesis we focus on information retrieval in documents whit explicit structure where extracting passages is not necessary for retrieving relevant text fragments to the user.

Remarkably, many documents on the web are more or less structured using one of the following formats: HTML, SGML, XML, etc. Additionally, many documents come with an already prepared classification or a short description of document content (e.g., keywords) in some of the tags. For example, some of the TREC [216] and CLEF [165] document collections are in SGML format with tags such as: “subject”, “section”, “type”. Furthermore, the INitiative for the Evaluation of XML Retrieval (INEX) provides the XML data (scientific IEEE collection) for the evaluation of XML retrieval approaches.

Having such setting, a number of structured IR systems, based on different retrieval models, have been developed in recent years. The most influential approaches, coming from the IR community, that tackle some or all of the identified structured retrieval aspects are discussed here. They are all adaptations of the flat text models presented in Section 2.1.1, and therefore keep most of their features and drawbacks. The database approaches that specify algebras for structured IR are explained in Section 2.4, while the approaches developed particularly for structured document search and retrieval are described in Section 2.5.

Adaptation of a vector space model for XML retrieval

Several extensions are proposed for adapting the vector space model to structured information retrieval [31, 51, 79, 81, 129]. The elementary assumption is that the document vector is now transformed into document component (element) vector. The relevance ranking is computed based on an *term-element frequency*, i.e., the number of occurrences of a term in a structured element, and *inverse element frequency*, i.e., the number of specific elements that contain the term. Due to different length and different distribution of elements in the structured collection, the results of the direct application of the vector space model to structured IR was not a success (e.g., see [5, 129]). An additional problem is nesting of elements present in, e.g., XML documents.

Component ranking with document pivot Mass and Mandelbrod [128] use different normalization function to handle distinct element length and distribution. The formula they use for ranked retrieval of elements is shown in Equation 2.15, where el is a search element, avg_{tcq} is the average number of occurrences of all query terms in a query q, avg_{tcel} is the average number of occurrences of all query terms in an element el, N_{el} is the number of elements el in the collection, $ec(tm_i, el)$ is the number of elements that contain term tm_i (element count), and $\|q\|$ and $\|el\|$ are query and element sizes used for normalization.

$$S(el,q) = \frac{\sum_{i=1}^{n} \frac{ln(tc(tm_i,q))}{avg_{tcq}} \cdot \frac{ln(tc(tm_i,el))}{avg_{tcel}} \cdot ln\left(\frac{N_{el}}{ec(tm_i,el)}\right)}{\|q\| \cdot \|el\|} \tag{2.15}$$

Furthermore, to include the evidence from the ‘surrounding’ elements in case of small sized elements, authors introduced a document pivot [129], similar to context smoothing in [202]. The relevance score ($S'(el,q)$) of an element, utilizing the relevance score of surrounding document doc, is now computed as depicted in Equation 2.16. The parameter dp needs to be estimated empirically.

$$S'(el,q) = dp \cdot S(el,q) + (1 - dp) \cdot S(doc,q) \tag{2.16}$$

XIRQL & augmentation weights Fuhr et al. [71] developed an XML retrieval system (called HyRex [79]) based on a tf.idf formula for term weighting and a path based algebra (see Section 2.4.2 for more details) that uses events and event probabilities to compute relevance scores for structured elements. The authors identified "atomic" units for which term weighting is computed. These atomic units (document components) are also the ones that can be retrieved as a result of the query evaluation.

As the goal of the system is to retrieve the most specific answer to the query, the authors introduced the concept of *augmentation weighting*: atomic unit relevance scores are downweighted, i.e., multiplied by an augmentation factor, when they are propagated upwards to the ancestor atomic unit. This concept is later followed in many structured retrieval systems as depicted below.

Vector Spaces The vector space approach to XML retrieval by Grabs and Schek [81, 82] followed the ideas on defining atomic units (called indexing nodes) introduced by Fuhr et al. [71]. In the retrieval process the authors differentiate between the single-category retrieval, multi-category retrieval, and nested retrieval. The term category denotes the set of structured elements that can be considered equivalent (e.g., section, subsection, abstract, and summary).

Single category retrieval describes the retrieval from indexing nodes in isolation. Equation 2.17 depicts the retrieval formula, where N_{cat} denotes the number of elements in the single category cat and $ef_{cat}(tm_i)$ denotes the element frequency of a term tm_i within the elements of cat.

$$S(el|q) = \sum_{i=1}^{n} \frac{tc(tm_i, el)}{len(doc)} \cdot \left(\log \frac{N_{cat}}{ef_{cat}(tm_i)}\right)^2 \cdot \frac{tc(tm_i, q)}{len(q)} \tag{2.17}$$

Multi-category retrieval describes the retrieval where structured elements belonging to different categories should be combined in the retrieval process. Multi-category retrieval is defined in Equation 2.18. The $\mathcal{M}$ denotes the set of indexing nodes in the multi-category.

$$S(el|q) = \sum_{i=1}^{n} \frac{tc(tm_i, el)}{len(doc)} \cdot \left(\log \frac{\sum_{cat \in \mathcal{M}} N_{cat}}{\sum_{cat \in \mathcal{M}} ef_{cat}(tm_i)}\right)^2 \cdot \frac{tc(tm_i, q)}{len(q)} \tag{2.18}$$

Finally, Equation 2.19 defines nested retrieval, i.e., retrieval where relevance from nested elements need to be combined. For this, the authors also followed the approach in [71] and used the augmentation weighting concept. The augmentation weight is denoted with $aw_\ell \in [0, 1]$ for each parent-child relation on the path $pc(el, el_j)$. e_j is an element that is inside the set of all elements contained in a sub-tree rooted at el (also including el) and denoted as $\mathcal{S}(el)$.

$$S(el|q) = \sum_{el_j \in \mathcal{S}(el)} \left(\prod_{\ell \in pc(el, el_j)} aw_\ell \cdot \sum_{i=1}^{n} \frac{tc(tm_i, el)}{len(doc)} \cdot \left(\log \frac{N_{cat}}{ef_{cat}(tm_i)}\right)^2 \cdot \frac{tc(tm_i, q)}{len(q)} \right) \tag{2.19}$$

Garden Point XML (GPX) The Garden Point XML (GPX) model presented by Geva [77] can be considered as a variation of a *tf.idf* approach. The basic formula in the GPX approach defines the relevance score of the XML element (el) with respect to query terms ($S(el|q)$) as:

$$S(el|q) = A^{a-1} \sum_{i=1}^{n} \frac{tc(tm_i, el)}{tc(tm_i, col)} \tag{2.20}$$

where A is the parameter with a value between 3 and 10, and a is a number of distinct query terms contained in an element el. The original formula defines the relevance score computation only for the leaf (XML) elements and not for the arbitrary elements in the collection [77]. The relevance score is then propagated to the elements higher in the hierarchy using Equation 2.21, where m is the number of children elements of a parent pel and $D(m)$ is a factor equal to 0.49 if $m = 1$ or to 0.99 if $m > 1$.

$$S(pel|q) = D(m) \cdot \sum_{i=1}^{m} P(el|q) \tag{2.21}$$

Bayesian model for structured retrieval

Inference (Bayesian) network models seem like an ideal framework for modeling structured document retrieval. Each element in the hierarchical organization of structured documents could be modeled as a node in the inference network. The inference process would then define how relevance scores are propagated through the network with respect to the parent or child nodes. However due to the complexity of the problem and exponential explosion of the size of conditional probability tables severe restrictions, i.e., canonical forms, must be used.

Multi-layered Bayesian network model Crestani et al. [47] developed a Bayesian network for structured retrieval based on a simple two-layered network. It consists of a term layer and document layer, where edges depict the dependence between the two. To keep the inference tractable they used a canonical model where each document node depends only on terms that are considered relevant for that document: $Re(pa(doc))$, as depicted in Equation 2.22. Here, $pa(doc)$ denotes the parent terms of a document node, w_{ij} is a term weight (i.e., a *tf.idf* like weight) defined such that $\sum_{tm_i \in Re(pa(doc))} w_{ij} \leq 1$, and $p(tm_i, q) = 1$ if $tm_i \in q$ or $p(tm_i, q) = 1/M$ if $tm_i \notin q$ (M denotes the number of terms in the collection).

$$P(doc|q) = \sum_{tm_i \in Re(pa(doc))} w_{ij} \cdot p(tm_i, q) \tag{2.22}$$

For structured retrieval the authors introduced a number of intermediate layers, each representing elements on a certain level in a structured document hierarchy. They kept the canonical model, now modeling the relations between elements in a document structure. Assuming the prior probability of term's presence: $p(tm_i) = 1/M$, and term's absence: $p(tm_i) = (M-1)/M$, and taking into account that all conditional probabilities are already specified using the Equation 2.22, a top-down inference can produce the relevance score for elements on arbitrary level in a document structure.

Multi-model Bayesian network Piwowarski et al. [170, 215] developed an approach similar to the one presented in [47] for modeling structured document retrieval. The main difference is that the network is built starting from the document root element and that the terms are excluded from the network. The relevance ranking of each element is computed based on parent (pel) relevance: $p(pel)$, and a number of baseline models m_i: $p(m_i|q)$. This is depicted in Equation 2.23. Baseline models define different aspects of retrieval models and are based on traditional IR models, such as *tf.idf* or Okapi. In [170], the authors used two models, one playing the role of a *tf* factor and the other of an *idf* factor.

$$P(el|q) = \sum_{m_1,m_2 \in \{rel,\neg rel\}} \theta_{c(el),el,pel,m_1,m_2} \cdot p(el|pel) \cdot p(m_1|q) \cdot p(m_2|q) \quad (2.23)$$

To keep the model tractable, relevance score computation follows some simple rules, e.g., a non relevant element cannot have a relevant descendant element. The network is trained to estimate the value of the parameter $\theta_{c(el),el,pel,m_1,m_2}$ that depends on the element category $c(el)$ (see the list in [170]), element name el, element name parent pel, and models m_1 and m_2 depending on their estimated relevance rel or non-relevance $\neg rel$. Based on this parameter and the simple rules, the relevance score of an element ($P(el|q)$) can be determined.

Language models for structured retrieval

The unigram generative language model theory is developed with the assumption that the terms in a document are independent and that the probability of generating a query from a document is independent of document structure. However, this is not the case for the structured IR. Two approaches exist for specifying language models for structured retrieval: (1) directly apply language models to element retrieval assuming element independence [96, 121, 202], (2) push the language model approach to the leaf element level and then develop a framework for combining/propagating the relevance score to the desired element [154, 155]. The second approach seems more appropriate for structured IR and is discussed below.

Hierarchical language models Ogilvie and Callan [154, 155, 156] developed a hierarchical language model to specify retrieval in structured documents. To compute the relevance of an element, the authors use its language model and combine it with parent and children language models. The model is based on formulas given in Equation 2.24. The first formula estimates the probability of generating the term (tm) given the element model (θ_{el}) for the element el, without taking into account its children or parent elements. It uses the flat text language model formula (see Equation 2.13). The background model can vary depending on the element name (type): $\theta_{type(el)}$ [156]. $P(tm|\theta'_{el})$ combines the language model of an element with language models of its children ($ch(el)$). Since the evaluation is done from the root of the tree down, the final equation computes the child language model based on a parent ($pa(el)$) language model. The parameters λ (λ^u – "universal", λ^c – child, and λ^p – parent parameters) are empirically estimated or computed based on a size of descendant elements [155].

$$\begin{aligned} P(tm|\theta_{el}) &= \lambda^u_{el} P(tm|el) + (1-\lambda^u_{el}) P\left(tm|\theta_{type(el)}\right) \\ P(tm|\theta'_{el}) &= \lambda^c_{el} P(tm|el) + \sum_{el_j \in ch(el)} \lambda^c_{el_j} P\left(tm|\theta'_{el_j}\right) \\ P(tm|\theta''_{el}) &= \lambda^p_{el} P\left(tm|\theta''_{pa(el)}\right) + (1-\lambda^p_{el}) P\left(tm|\theta'_{el}\right) \end{aligned} \tag{2.24}$$

Finally, the relevance score of an arbitrary element in the collection is computed as shown in Equation 2.25.

$$P(q|\theta''_{el}) = \prod_{i=1}^{n} P(tm_i|\theta''_{el})^{\frac{tc(tm_i,q)}{len(q)}} \tag{2.25}$$

Discussion

All structured retrieval models described previously have roots in flat text retrieval models. However, they are usually more complex and contain more parameters that need to be empirically estimated. They all try to model the dependence present in structured documents, without explicitly stating what are the elementary aspects that need to be addressed when modeling structured retrieval. We argue that detailed analysis of the structured retrieval problem is necessary. The analysis of structured retrieval would help us to better understand the problem and to identify the elementary structured retrieval requirements. Then, each retrieval requirement could be modeled independently, and the best combination of these models could be found. This is the approach that we take and it is described in Chapter 3.

2.1.3 Retrieval models for multimedia documents

Multimedia documents contain multiple types of data, namely text, video, images, and music (audio). Multimedia retrieval systems retrieve information from various information sources and try to fuse them in a ranked list of relevant multimedia data, i.e., video shots, images, songs. While retrieval models in the textual domain, both structured and unstructured, show gratifying effectiveness, retrieval models developed particularly for video, image, or audio search are by far less effective, and in most cases need to be combined with text search [218]. The exceptions are domain specific retrieval, where domain knowledge is used to improve the search (e.g., [166]), or speech recognition [157].

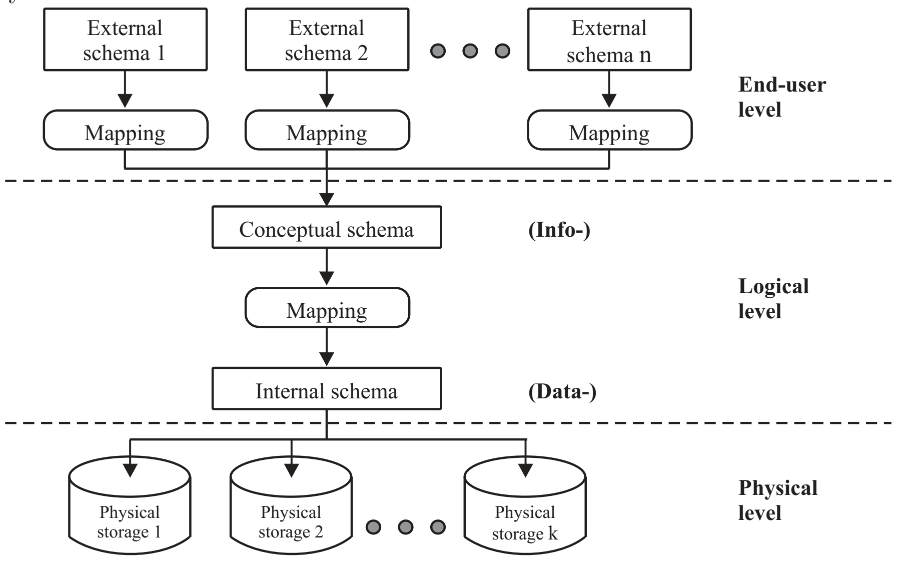
Unlike in text retrieval, where elementary units for specifying retrieval models are terms, in image and video retrieval such units cannot be identified. For example, one pixel of an image or a video, or a pitch of a sound in a specific instance of time, cannot be considered as an elementary unit for retrieval. Much research has been done (e.g., [13, 166]) in specifying low-level features, such as dominant color, edges, texture, and shape in an image, pitch in a sound, motion flow in a video, that are able to describe the basic units of multimedia data. However, such data extracted from images and videos cannot be directly used to describe the content of a document. In other words, crossing the semantic gap between this low-level features and semantic concepts that describe the image/video content is still an open issue in the multimedia IR area.

To be able to search non-text documents, the content of multimedia documents needs to be identified, i.e., *content description* is needed. The approaches dealing with automatic content extraction are termed *content-based* multimedia retrieval approaches. For example, Petković [166] presents an approach that uses hidden Markov models to detect the strokes in a tennis match, based on a low level visual features that describe the player (mass center and position of the arms), and a simple algebra to describe the events in a tennis match. Furthermore, he uses dynamic Bayesian network to detect highlights in Formula 1 races based on a low-level audio (pitch, short term energy, etc.) and video (color histogram and shape) features, and combine this information with the superimposed text extracted from the screen, to describe interesting events, such as fly-outs or overtaking, during the race.

In content-based retrieval approaches, a significant portion uses the query by example paradigm [13]. Query by example retrieval models either use low-level features such as color, motion, pitch of a sound [13, 166, 179, 217], or use high-level features that can be extracted from multimedia documents, such as car, explosion, building, face [33, 36, 152]. However, the effectiveness of these retrieval models on their own is not sufficient, and the fusion with text search based on automatically/manually annotated multimedia data is needed.

Content-based video retrieval is also addressed in this thesis, but only a specific part where text retrieval approaches are applied to multimedia retrieval and where relevance results of different multimedia retrieval techniques need to be fused in

Figure 2.2: The layered DBMS architecture consisting of end-user, logical, and physical levels.



one framework. We focus only on how content extracted from the source data, in the form of annotated documents (using, e.g., XML XMT MPEG 7/4/1 format [127]), can be used, and how query by example similarity search results can be fused with the text search for the effective multimedia information retrieval (see e.g., [94, 104, 217] for an overview).

2.2 Closing the gap between databases and IR

This section starts with discussing the database approach and revealing the problems in the integration of databases and information retrieval systems. It then emphasizes the conceptual differences between databases and IR systems focusing on the logical algebra. The section ends with presenting several most influential approaches that integrate database and IR systems.

2.2.1 The database approach

The main characteristic of the database approach is its layered architecture with a strong separation between *external*, *conceptual*, and *internal* schema, as described in the introduction of this thesis. From a DBMS designers point of view [2, 92] these schemas are defined at the *end-user*, *logical*, and *physical* levels of a database, as depicted in Figure 2.2. In the figure, the logical level consists of two sub-levels

[200]. The top one, called infological level, defines a framework for implementing the conceptual schema, while the bottom one, called datalogical level, is more related to the data manipulation using the internal schema and the physical storage structures.

The layered architecture provides *data independence*, consisting of *logical* and *physical* data independence. Logical data independence is the possibility to change the conceptual schema without having to change the external schemes, and physical data independence is the possibility to change the internal schema without having to change the conceptual schema. The introduction of the conceptual schema at the logical level that isolates end-user representations and operations from physical representations and operations, characterizes the database approach.

2.2.2 Databases & IR

Despite the numerous existing systems dealing with structured querying, the problem of expressing as well as executing IR-like queries using (relational) databases is still an open issue [34]. An IR-like query does not specify hard conditions on documents (or document components), but queries the collection for documents 'about' a certain topic. For instance, an XML element that is relevant to a query for elements about "beautiful flowers" might not contain the phrase "beautiful flowers", or even both words "beautiful" and "flowers". IR-like queries should result in a ranked list of answer elements, in decreasing order of some score value that the system assigns to each element. The score value has to reflect the probability (or degree) of relevance of the element to the IR-like query.

DBMSs have trouble supporting such IR tasks. They are based on a layered architecture depicted in Figure 2.2, supporting data independence (as well as content independence) by having an external, conceptual, and internal schema. For (structured) IR systems, following this separation in levels, would give another, additional advantage over flat file IR systems. This is reflected in enabling *content independence* [55]: the independence between the development of the application programs that access the content using the external schema and the actual document representation using the internal schema.

Following this reasoning, the central component of a DBMS that needs to provide the logical and physical data independence when implementing a database is the conceptual schema. It is usually described in terms of a mathematical framework (logical algebra) for manipulating the domain of values in the data model. In relational DBMSs this role is played by the relational data model and relational algebra [40]. Therefore, the most important component that needs to be developed, altered, or extended, when fully integrating IR systems with a DBMS is the algebra (see, e.g., [75, 196]). As relational algebra is still predominant in modern DBMSs we shed some thoughts on it before discussing systems that integrate IR and DB.

Algebra

An algebra is a formal framework for data manipulation based on *operators* and a *domain* of values. The operators map values taken from the domain into other values in the domain. If we take as an example the domain of all integers, and as operators we define sum '+' and product '$\cdot$', then examples of algebraic expressions are: $1+2$, $3\cdot4$, or $1+2\cdot3$. The results of these expressions are again in the domain of integers, i.e., we say that the operators '+' and '$\cdot$' are *closed* in the domain of integers.

A well-known algebra for data manipulation in database management systems is the *relational algebra* [40]. The domain of the relational algebra consists of *relations* usually represented as *relational tables.* A relation is a sets of *tuples*, where each tuple is a finite sequence of objects, i.e., *attributes.*

The set of primitive (basic) operators in relational algebra consists of the following six operators: selection $\sigma_{a\theta x}(R)$, projection $\pi_{a_1,a_2,...,a_n}(R)$, Cartesian (cross) product $R_1 \times R_2$, set union $R_1 \cup R_2$, set difference $R_1 \setminus R_2$, and rename $\rho_{a/b}$. Here, a_i, $i = 1, ..., n$, and b are attribute names, θ is a binary operator in the set $\{<, >, =, \leq, \geq\}$, x is either an attribute name or a constant value, and R_1 and R_2 are relations.

Based on primitive operators many operators are derived, such as set intersection ($R_1 \cap R_2$) and natural join ($R_1 \bowtie R_2$). All these operators are based on operators in set theory, but are actually a subset of the first-order logic. Operators in the relational algebra are Boolean in nature. In other words, each tuple in the relation either satisfies the condition in the operator or not. Therefore, knowing the original relational data set, using relational expressions, the output is always deterministic. There is no notion of probability or relevance.

SQL

Relational algebra is a low-level, operator-oriented language. Creating a query in relational algebra involves combining relational operators using algebraic notation. On the other hand, the relational model defines another 'language' for accessing relational databases – relational calculus. Relational calculus is a high-level, declarative language. Creating a query in relational calculus involves describing what results are desired from a database.

A version of a relational calculus is Structured Query Language (SQL) [32, 40]. It is a declarative computer language used to create, modify, retrieve and manipulate data from relational database management systems. It is an ANSI (American National Standards Institute) and ISO (International Organization for Standardization) standard. SQL is currently the most common query language used as an interface to databases.

There are additional features in SQL apart from those that merely implement features of relational algebra or calculus, such as arithmetic operators, aggregate functions, as well as support for data insertion, modification and deletion. As its syntax is rather complex and as it is not of importance for the thesis we do not discuss it. For more details on SQL we refer to any database book (e.g., [112]).

Algebraic properties

Algebra operators usually have certain properties. For instance, the arithmetic multiplication operator ('$\cdot$') distributes over arithmetic addition operator ('+'), i.e., $(1 + 2) \cdot 3 = (1 \cdot 3) + (2 \cdot 3)$. Similarly, based on the definition of relational operators [40], many properties of relational operators hold. For example, set intersection distributes over set union:

$$\begin{aligned}(\sigma_{a_1\theta x_1}(R_1) \cap \sigma_{a_2\theta x_2}(R_2)) \cup \sigma_{a_3\theta x_3}(R_3) &= \\ (\sigma_{a_1\theta x_1}(R_1) \cup \sigma_{a_3\theta x_3}(R_3)) &\cap (\sigma_{a_2\theta x_2}(R_2) \cup \sigma_{a_3\theta x_3}(R_3))\end{aligned} \tag{2.26}$$

This and many other properties of relational algebra illustrate that there are many expressions that give exactly the same results. However, some expressions might require more processing power and more memory for intermediate results when executed in a DB system (at the physical level). Avoiding expressions that require long processing time, using algebraic operator properties, is called *logical query optimization* in (relational) database systems [16, 100, 120].

Looking from the IR perspective, properties like Property 2.26 might be important for another reason. When extending algebras to enable ranked retrieval, it might happen that expressions that give exactly the same matching results, are producing different document rankings. If we base a query optimizer on the properties of the original algebra, then the system would produce different rankings depending on the query plan chosen. The extended Boolean models [193] and fuzzy set models [160] (presented in Section 2.1.1) might for instance show this kind of unpredictable behavior. This issue is important when developing an algebra for IR, and is revisited in Chapter 4 but from the perspective of score region algebra.

2.2.3 Approaches for DB & IR integration

Looking from the perspective of logical algebra we can conclude that the integration of information retrieval and database management systems can be achieved in two ways, one without altering the algebra and the other with altering it. Therefore, we have two types of integrated systems [214]:

1. *loosely-coupled* systems, and
2. *tightly-coupled* systems.

The integration in loosely-coupled systems is done at the higher (end-user) level, without altering the core of a DBMS, i.e., logical and physical level. On the other hand, tightly-coupled systems integrate the IR aspects along end-user and logical levels in a three-level DBMS, or reimplement the database system at all three levels.

Loosely-coupled systems

In loosely-coupled integration, an IR system can be implemented as a *hybrid* system, viewed as an *application* of a DBMS, or specified as a *query language extension.*

Hybrid approach In the hybrid approach, IR and DB systems are kept apart and the results are combined at the application level. For example, in [214] the Inquery retrieval system [30] is used to output the information about ranked textual data which is sent to Sybase DBMS as an SQL query that searches the corresponding data in a database. Another hybrid approach in the structured retrieval domain that uses full-text search engine Zettair[1] and a native XML database eXist[2] [133] for XML retrieval is presented in [163]. Although systems like these are easy to realize, they are inflexible for combining IR and Boolean parts of the query. Furthermore, all the 'hard work' is done at the application level.

IR as an application of a DBMS In this approach the complete IR system is implemented using database features. In relational databases that would mean that the retrieval function is implemented using SQL query statements. This results in a high complexity of query expressions that are difficult to handle. Additionally, it is quite cumbersome for the user to specify the query unless a high-level application is used. However, such systems are easy to implement and quite good for prototyping. Examples of such systems are PowerDB-IR [80] where ranking algorithms are realized on top of a database cluster, and the system developed by Grossman et al. [83, 84, 85] where authors implemented relevance ranking using plain SQL.

Query language extension IR models can be implemented as an extension of existing database query languages, e.g., SQL. In [214] authors developed an extended SQL (ESQL) syntax that is based on Inquery query language. Actually, the Inquery query is specified as a special text component of the SQL syntax, i.e., the TEXT_QUERY clause in the *WHERE* part of SQL. The text component of the ESQL queries is transformed into SQL and executed using an of-the-shelf DBMS. This approach is quite similar to the previous one except that the IR search is formally expressed using a query language instead of using a high-level ranking algorithm.

[1]http://www.seg.rmit.edu.au/zettair/
[2]http://exist.sourceforge.net/

Tightly-coupled systems

We can distinguish two approaches for developing tightly coupled systems. The first one is based on altering the logical algebra and eventually end-user query language (SQL) of a database – *algebraic extensions*. The second one demands a reimplementation of a database system at all levels to handle ranked retrieval – *full integration*.

Algebraic extension IR systems can be integrated more tightly with relational databases in the sense that the logical algebra is altered to support information retrieval operators. This is the approach taken by Fuhr and Rölleke [68, 69, 75]. The authors propose a generalization of a relational model and introduce operators for handling uncertainty. The algebra is called probabilistic relational algebra. Probabilistic IR models are used as a base for modeling uncertainty operators in the algebra.

Instead of relations and relational operators, the authors define events and event probabilities. Each event corresponds to a relation, extended with the attribute representing the event probability. The six basic relational algebra operators are redefined such that besides attribute manipulation they form a Boolean combination of event expressions. The evaluation of the Boolean combination of event expressions results in the relevance score of an event (relation).

Full integration The first paper to describe the full integration of databases and information retrieval systems is published by Schek and Pistor [196]. Authors explained the complete development of an integrated system along three database levels. The model is based on an extension of the relational model that permits nesting of relations, called non-first normal form (NF^2) relations. The authors introduced *nest* and *unnest* operators to handle nested relations and *lists* to cope with ordered sets of items. They extended the SQL syntax for handling new non-atomic attributes (that contain nested relations) and specified operators for field oriented ranked retrieval. At the physical level they extended the traditional database indexing techniques with new ones for storing and retrieving terms. This resulted in the Darmstadt Database Kernel System [53].

The SPIDER system [195] was another approach for the integration of DBMS and IR systems. It is based on a model that defines a hierarchy of heterogeneous algebras, whereas the retrieval model is based on a probabilistic model. The authors developed a completely new query language called FQL for expressing the queries.

The probabilistic relational algebra, explained above as an algebraic extension approach, was a starting point for developing probabilistic datalog [184] – a logic that enables uncertain inference for describing the retrieval process. Probabilistic datalog was later employed for developing a database system used for hypermedia retrieval applications consisting of interlinked multimedia data [116].

With the proliferation of multimedia and structured documents, hypermedia information retrieval systems became a precious ground for the integration of IR and DBMS systems. The Mirror system [56, 59, 60] is one that demonstrates the full integration of a multimedia retrieval system and a DBMS. It uses the Moa object algebra [212] at the logical level and parallel main-memory database system MonetDB [19] at the physical level. Moa is extended to support document representations and ranking, and extended with operators for computing term statistics. Moa logical expressions are transformed into sequences of operators in Monet Interpreter Language (MIL), which uses an algebra for the binary relational data model, supported by the MonetDB kernel. The MonetDB relational data model operates on flattened representations of the content in the form of binary tables (BATs). The Moa algebra expression that specifies an IR request is implemented using the standard MonetDB operator set extended with new operators for computing relevance scores using information stored in BATs [56].

2.3 Structured documents and query languages

For illustrating structured documents we choose XML. XML [22] is the de-facto standard for the storage and exchange of documents in the structured format nowadays. Furthermore, XML is derived from SGML [54], from which the HTML [174] is also derived, and therefore bears a great similarity to web documents. Structured querying is explained using XML query language data models.

2.3.1 XML data model

One can think of an XML document as a linearization of a tree structure [21] (see Figure 2.3 introduced later in this chapter), where nodes represent either character string depicting the true document content or meta information, and edges represent parent-child relation. However, connections between arbitrary nodes in a tree structure of XML can be made, using e.g., *IDREF* attributes, which makes it a graph. The semantics of these edges is different from the ones that define the XML tree structure, and we call them *hyperlinks*. They are either XPointers [61] that support addressing internal structure of XML documents or XLinks [62] which allow links to external XML documents.

The XML data model assumes six node types: (1) document, (2) element, (3) attribute, (4) processing instruction, (5) comment, and (6) text. In the definition of the simple XML data model we disregard the namespaces and document type declaration (a link to a Document Type Definition – DTD), since they do not influence the generic XML data model.

Document node Only one document node exist in each XML document. It specifies some properties of a document and defines the root document node. The document node consists of a prolog which contains an XML declaration (something like ⟨?xml version = "1.0" encoding = "UTF-8"?⟩), document type declaration (like ⟨!DOCTYPE greeting SYSTEM "hello.dtd"⟩), root element node, and zero or more comment and processing instructions nodes in between. The document node must have a declaration and an element node; if not, the XML document is considered not well formed.

Element node Element node can be defined as $en := (nn, A, CE)$. Here nn is an element node name, i.e. a compound string (name string)[3] used to denote the node. The presence of a name is required for each element node. A represents an attribute set consisting of pairs (an, av), where an stands for attribute name, and av for corresponding attribute value. Both, attribute names and values, are (compound and reference) strings [22]. Attributes can be of a predefined attribute type, such as '*ID*' which is used as a unique identifier of an element node. The unique identifier can be used for making a hyperlink from an arbitrary node in an XML tree, e.g., by using the predefined attribute type '*IDREF*'.

CE is an ordered sequence of child nodes (ce) which can be of any node type except document node type or an attribute. The sequence CE can also be an empty sequence in case an element node has no child nodes. Since an XML document must start with an element node, called root node, the whole XML tree can be defined recursively using tuples (nn_i, A_i, CE_i) starting from (nn_r, A_r, CE_r), where nn_r is a root node name, A_r is a set of root node attributes and CE_r is a ordered set of root child nodes.

Processing instruction node Processing instructions are composed of the target or the name of the processing instruction (pn) and the data or information for processing an XML document or element (data): ⟨?pn data?⟩. Processing instructions, as well as comment and text nodes, cannot contain any children nodes.

Comment node Comments are defined in XML syntax as ⟨! -- com -- ⟩, where com is a character string.

Text node Text node is always a leaf node. This is the only node for which it is not required to use delimiters for the specification[4]. The text node can contain only character strings.

2.3.2 XML data models for querying structured data

Although the basic XML data model is sufficient for describing XML structure it is not appropriate for XML tree traversal. Therefore, this model had to be modified to support XML query languages, such as XPath [14, 37] and XQuery [18].

[3]Compound string is formed using the restrictions on lexical structure according to [22]. The most important restriction is that the string must be formed without the usage of white spaces.

[4]However, these delimiters can be specified using the CDATA sections in the next manner ⟨![CDATA[string]]⟩.

XPath 1.0 data model

The XPath 1.0 [37] data model defines a tree model against which all XPath expressions are evaluated. Most expressions are used to navigate the XML tree model. These expressions are named location paths and the result of an application of a location path on an XML tree is a *node-set*. The XML tree traversal is performed in document order which is defined as a pre-order traversal of XML tree data model. However, the XPath 1.0 data model does not specify how the XML tree (logical structure) is constructed, i.e., it is implementation dependent [203].

The tree traversal in XPath is specified as a location path consisting of consecutive steps: $loc\_path := /step_1/step_2/.../step_n$, where each step has the form of $step := axis::nodetest[predicate]$. XPath uses thirteen axis steps: self, parent, child, ancestor, ancestor-or-self, descendant, descendant-or-self, following, following-sibling, preceding, preceding-sibling, attribute, and namespace, as well as node tests on name and type. Predicates are Boolean expressions that test the values of location paths specified in a predicate (see [37] for detailed description).

Except node-sets, the XPath 1.0 data model supports three additional data types, namely Boolean, number, and string, which are actually the results of the XPath expression evaluation (within a predicate). To be able to compare node-sets with entities of other types, node-sets are mapped to a string value. Each node type identified in XPath data model has a specific definition of a string value. It includes two awkward definitions of element and root node string values as a concatenation of all descendant text nodes. Although these definitions are reasonable and make easier conversion and comparison of node-sets with entities of other types, it becomes problematic when evaluation of expressions including string manipulation is considered.

The problems imposed by this definition of string values are illustrated using three examples. First, if we want to evaluate some expression that include string comparison on a root node (or nodes which are high in hierarchical structure of XML) we have to perform comparison with respect to the contents of all descendant text nodes, which in case of large XML documents can be very difficult.

The second problem is concerned with the string value construction. If we consider the following XML name element:

⟨name⟩⟨fname⟩John⟨/fname⟩⟨sname⟩Smith⟨/sname⟩⟨name⟩

the string value of `name` element will be '*JohnSmith*'. This is also undesirable property for string comparison.

The third problem concerns the definition of functions for node string value comparison with a constant value. For example, the result of applying expression `child::item[value>=100]` in XPath 1.0 will yield different results based on the specification of the *value* element node. If the *value* node is defined in the form of ⟨value⟩$200⟨/value⟩, the node will not be in the result node-set since the coercion of the string value *$200* is *NaN* (not a number). Similarly, node defined as ⟨value⟩200$⟨/value⟩ will not be returned as the numeric value is not followed with a white space, as defined in [37].

Table 2.1: Node types and their properties in XPath 2.0 and XQuery 1.0 data model.

Node type	Node type properties
Document node	{*base-uri, children, unparsed-entities, document-uri*}
Element node	{*base-uri*, ***node-name***, *parent*, ***type***, *children, attributes, namespaces*, ***nilled***}
Attribute node	{***node-name***, ***string-value***, *parent*, ***type***}
Namespace node	{*prefix*, ***uri***, *parent*}
Proc. instruction node	{***target***, ***content***, *base-uri, parent*}
Comment node	{***content***, ***parent***}
Text node	{***content***, ***parent***}

XPath 2.0 & XQuery 1.0 data models

In order to overcome some of the drawbacks of XPath 1.0 data model built upon node sets, the World Wide Web Consortium (W3C) proposed a new XML data model. This data model defines the XML information content using the XML *information set* [46]. The XML information set consists of 11 *information items* (node types) where each information item has many new *properties*.

The main difference between the node-set data model and the information item data model is the absence of the text node, i.e. the text information item. Instead, a character information item is defined. The character information item is introduced to enable reasoning about the white spaces and content words in an XML document. It can be a character code – a code of the character as defined in ISO 10646[5], element content whitespace – a Boolean value indicating whether the character is white space, and parent referring to the information item containing the character information item in its children property .

The data model is defined over a *sequence* (ordered collection) of zero or more *items*, while the items are either *nodes* or *atomic values*. An atomic value is a value in the value space of an atomic type. An atomic type is one of the simple atomic types, such as integer, string, date, Boolean, or a type derived by restriction from another simple atomic type, as described in [15]. The XPath 2.0 and XQuery 1.0 data model preserves the basic node-set framework from the XPath 1.0 data model, now called the item sequence framework. It extends the node-set framework with a number of *properties*. It also adds an extra entity in the model, named atomic type, which can form a sequence as well. Node types used in the data model are presented in Table 2.1, along with their properties (*uri* stands for universal resource identifier). Node properties written in bold are required for the definition of a specific type of node.

The XPath 2.0 [14] and XQuery 1.0 [18] data models contain a *typed-value accessor* in order to overcome the problem of string-value definition [65]. The

[5]http://std.dkuug.dk/jtc1/sc2/wg2/

typed-value accessor returns the content of nodes (the string values of all the leaf element nodes) if they are not of a complex type with a complex content, i.e., a mixture of atomic types. As XML documents are graph-structured, the data model is defined using conventional terminology for graphs. The data model is a node-labeled, directed graph, in which each node has a unique identity[6]. Unlike in XPath 1.0 a document order is defined among all the nodes in an XML graph.

Discussion

As can be seen, the original XML as well as XPath and XQuery data models are not very supportive for evaluating full-text search. First, data models represent element content as string values and therefore impose the string manipulation as only way to perform full-text search. Second, it is difficult to explore element content in terms of single words and their statistics in XML elements if they are represented as part of a string.

2.4 Structured IR and databases

In structured IR, the user can specify not only his information need, but also where to search for information. Many structured (XML) IR query languages use the W3C[7] query languages (XQuery [18] or XPath [14, 37]) as a starting point and extend them with IR-like search expressions. Typical examples are full-text search extension of XQuery [7], Narrowed Extended XPath I (NEXI) [209], and XIRQL [71]. They enable the distinction between *search elements* and *answer elements* in structured IR, where both element types are not predefined as in flat text IR (documents). Search elements are elements where the user searches for specific information, while answer elements are elements that the user would like to obtain as an answer to a query.

The retrieval models implemented in traditional IR systems typically represent either one class of retrieval models (e.g., like language models in Lemur [155]) or a single retrieval model (e.g., inference network in Inquery [30]). These models are, however, not expressive enough to model complex document component retrieval consisting of more retrieval subtasks, modeling different retrieval aspects. Their application to structured retrieval also results in the explosion of parameters that need to be estimated (as can be seen in Section 2.1.2). Furthermore, these models are often used to model retrieval subtasks, introducing the problem of combining these sub-models in one model, where this combination requires a systematic approach. Therefore, the simple one-model search engine concept has to be replaced with the support for more advanced complex structured retrieval

[6]This id is not the same concept as ID used for representing references in ID/IDREF correlation.
[7]http://www.w3.org/.

models that either extend flat text retrieval models [47, 77, 128, 155, 170] or support the combination of different flat text retrieval models [5].

With respect to the physical implementation, there is a consensus of using a variant of the inverted index structures [90, 226] for IR systems. However, using inverted index structures for structured retrieval would lead to having a large redundant storage of "inverted elements" for each term-element pair in a document. This implies that new indexing techniques have to be developed to cope with this new IR task, either as a variant of the inverted index structures [79, 128, 155] or by using database indexing facilities [66, 77, 164].

2.4.1 A relational view on XML

Most database approaches for structured document manipulation choose to explicitly define an index when modeling structured documents. Index is used as an identifier for components in an XML document and in most cases also conveys the information about the structured organization of XML documents. The rationale for using such indexes are benefits that can be achieved when querying such indexed relational representation of structured documents. An indexed XML document, however, is not modeled as one relational table since this table would be huge and in most cases (on most platforms) hard to process. In many relational approaches to XML different fragmentations of this basic table are used. We can identify three broad approaches in transforming the XML data model to relational data model:

1. horizontal fragmentation based only on type of XML nodes [88, 122]
2. vertical fragmentation based on a name and/or type of XML elements [26, 66]
3. path based fragmentation based on paths to XML nodes in an XML tree structure [77, 177].

The distinction is based on the definition of relational tables in the relational model. We give a short description of all three approaches.

Horizontal fragmentation

For horizontal fragmentation, the data set creation, i.e., the formation of the initial data set from (plain text) XML documents, can be explained through the usage of a two step indexing process. The indexing process is explained using an example XML document given in Figure 2.3. In the first step each token in the XML document – D, is indexed regarding its relative position with respect to the beginning of a document and its type ($I_1 : D \rightarrow X$). As a result a set of entities is obtained – $x \in X$, uniquely identified by their position in the XML document. Each entity has the form of $\{position, token, token\_type\}$, e.g., the 'story' tag is represented as $\{0, story, node\}$, attribute name 'src' as $\{13, src, attr\_name\}$, and term 'Rock' as $\{16, Rock, word\}$.

Figure 2.3: Example XML document.

```
<story id=``78''>
  <title>The Selfish Giant</title>
  <author>Oscar Wilde</author>
  <image src=``Garden.jpg''>
    <title>Rock garden</title>
  </image>
  <p>Every afternoon, as they were coming from school, the children used to go ... </p>
  <p>It was a large lovely garden, with soft green grass.  Here and there over ...  </p>
  <p>One day the Giant came back.  He had been to visit his friend the Cornish ... </p>
   ...
  <p>And when the children ran in that afternoon, they found the Giant lying dead under
    the tree, all covered with white blossoms.</p>
</story>
```

Table 2.2: Relational representation of XML document presented in Figure 2.3 obtained after the composition of initial indexing (I_1) and final indexing (I_2).

start	end	name	type
0	1067	story	node
1	2	id	attr_name
2	2	78	attr_value
3	7	title	node
4	6	-	text
4	4	The	word
5	5	Selfish	word
6	6	Giant	word
...	...	...	...
12	19	image	node
13	14	src	attr_name
14	14	Garden.jpg	attr_value
15	18	title	node
16	17	-	text
16	16	Rock	word
17	17	garden	word
...	...	...	...

Table 2.3: Relational representation for storing example XML document presented in Figure 2.3 after horizontal fragmentation of Table 2.2.

Node table $\mathcal{N}$

start	end	name	type
0	1067	story	node
3	7	title	node
4	6	-	text
...	...	...	...
12	19	image	node
15	18	title	node
16	17	-	text
...	...	...	...
46	188	p	node
47	187	-	text
...	...	...	...
1043	1066	p	node
1044	1065	-	text

Word table $\mathcal{W}$

start	name
4	The
5	Selfish
6	Giant
...	...
51	lovely
52	garden
...	...

Attribute table $\mathcal{A}$

start	owner	name	type
1	0	id	name
2	0	78	value
13	12	src	name
14	12	Garden.jpg	value

The second step produces regions that we can consider as the basic elements in the (relational) data model. These regions are produced by pairing corresponding tokens that represent opening and closing tags, attribute names and values, etc., and by removing markup delimiters from the tokens ($I_2 : X \rightarrow R$). This results in a data set like the one presented in Table 2.2. Thus, the initial data set construction can be defined as a composition of two indexing procedures: $I = I_1 \circ I_2(D)$. Although the indexing is a two step process it can be implemented as a single walk through an XML document using the SAX parser and stack structures (see [88]).

By applying horizontal fragmentation on such indexed XML data, different relational tables can be defined for the XML element nodes and attribute nodes. Additionally, the word table can be specified as a separate one that models the words in the XML text nodes. This is depicted in Table 2.3. Such tables can be used for efficient querying of XML documents [87].

Vertical fragmentation

Table 2.2 can also be partitioned for each pair of *name* and *type* in the name-type relational tables [66][8]. These tables, assuming that the parent element start attribute is used in addition to attributes depicted in Table 2.2, would have the form $name\_type(start, end, parent\_start)$. For illustration, the paragraph-element ta-

[8]In [66] different partitioning are described in the scope of specific query execution speedup.

ble would look like $p\_element = \{(46, 188, 0),\ \ldots\ , (1043, 1066, 0)\}$, and the garden-term table would look like $garden\_term = \{(17, 17, 15), (52, 52, 46), \ldots\}$.

The reason for partitioning the table in such a way is to avoid numerous joins in the execution of queries on terms occurring in a single element or path-like query expressions. However, the performance of the database system that uses this kind of partitioning shows difficulties for handling XML data in relational databases [66]: (1) storage inefficiency – the size of the relational database for XML file was an order of magnitude larger than the size of the original XML file, (2) long loading time – 27 minutes for 7.7MB file, and (3) long query execution times – $1s$ to $10s$ on 7.7MB file.

Path based fragmentation

Path based fragmentation can be also considered as a special version of the horizontal fragmentation. The difference is that in the indexing process the absolute path to every XML node is kept in a relational table. This path is used to make numerous relational tables. Each table consists of a set of relations that describe all the XML nodes that can be accessed by the same path in a document (collection) [199]. For example, the paragraph nodes (p) in the example XML document depicted in Figure 2.3 will all be in the same relational table that might look like $\{\ldots, (46, 188, node), (1043, 1066, node), \ldots\}$. To keep the pointers to different node tables, additional path summary table needs to be defined.

This fragmentation method might be efficient if XML queries are formulated using absolute paths that utilize the complete path specification (using parent/child relations). However, as shown by Ramírez and de Vries [177], it is inefficient if queries are formulated using descendant/ancestor relationships. Furthermore, the authors show that combining two relational schemas, i.e., maintaining two redundant data storage structures at the physical level, and using smart physical query optimization, faster query execution times can be achieved.

Discussion

A promising approach to executing structured (XPath and XQuery) queries is the use of relational database technology [86, 224], which can be extended to IR-like querying of structured documents [66, 77, 82, 198]. However, the semantics of XPath and XQuery give rules for navigation through XML structure, but not the rules that specify how score values for XML elements should propagate and relate to each other (as discussed in Section 4.2). Similarly, the semantics of relational algebra introduce rules for manipulating relational tables that describe XML data, but the rules for score computation and propagation cannot be derived from the relations present in the relational database. Therefore, relational algebra needs to be adjusted to support ranked retrieval in structured documents.

To illustrate this we use the following example NEXI query, assuming that the *about* clause should be followed strictly (similar to the XPath *contains* statement):

```
//story[about(.//p, beautiful flowers)
      and about(.//p, children)]
   //image[about(., garden) or about(., flowers)]
```

For the chosen storage model, e.g., when using horizontal fragmentation composed of the tables $\mathcal{N}$, $\mathcal{W}$, and $\mathcal{A}$ (see Table 2.3), we can directly transform any NEXI (XPath) expression into relational algebra expression. For the NEXI example query a possible relational algebra expressions could be specified as given in Equation 2.27.

$$\begin{aligned}
R_1 &= \sigma_{name=\text{'}story\text{'}}(\mathcal{N}),\ R_2 = \sigma_{name=\text{'}p\text{'}}(\mathcal{N}),\ R_3 = \sigma_{name=\text{'}image\text{'}}(\mathcal{N}) \\
R_4 &= \sigma_{name=\text{'}beautiful\text{'}}(\mathcal{W}),\ R_5 = \sigma_{name=\text{'}flowers\text{'}}(\mathcal{W}) \\
R_6 &= \sigma_{name=\text{'}children\text{'}}(\mathcal{W}),\ R_7 = \sigma_{name=\text{'}garden\text{'}}(\mathcal{W}) \qquad (2.27) \\
R_8 &= (R_1 \bowtie_{\sqsupset} ((R_2 \bowtie_{\sqsupset} R_4) \cap (R_2 \bowtie_{\sqsupset} R_5))) \cap (R_1 \bowtie_{\sqsupset} (R_2 \bowtie_{\sqsupset} R_4)) \\
R_9 &= ((R_3 \bowtie_{\sqsupset} R_7) \cup (R_3 \bowtie_{\sqsupset} R_5)) \bowtie_{\sqsubset} R_8
\end{aligned}$$

In the expressions we replaced a group of expressions consisting of join and projection operations, that simulates the XPath descendant/ancestor step, with special joins – $\bowtie_{\sqsupset}$ and $\bowtie_{\sqsubset}$, defined in Equations 2.28 and 2.29.

$$R_i \bowtie_{\sqsupset} R_j = \pi_{start_i,end_i,name_i}(R_i \bowtie_{start_i>start_j,end_i<end_j} R_j) \qquad (2.28)$$

$$R_i \bowtie_{\sqsubset} R_j = \pi_{start_i,end_i,name_i}(R_i \bowtie_{start_i<start_j,end_i>end_j} R_j) \qquad (2.29)$$

As can be seen, Equations 2.28 and 2.29 define the most frequently used expressions in the example query. This group of expressions actually represents the bottleneck for the XPath (NEXI) query processing, since its naive execution is extremely slow. A number of techniques have been proposed to speed up the execution of XPath descendant and ancestor steps, such as multi-predicate merge join [224] and staircase join [88]. Using such specialized join operators, the query execution becomes more efficient. However, these techniques also illustrate the problem of expressing structured document search using relational algebra.

Furthermore, the problem enlarges when IR tasks come into play. IR search asks not only for determining the containment relations among elements and terms, but it asks also for computing various term statistics. This cannot be easily expressed in relational algebra, which might imply that a new algebra is needed at the logical level of a database for specifying structured IR. The main reasons for introducing a new algebra for modeling structured information retrieval are discussed in Section 3.2.1 in the next chapter.

2.4.2 XML IR algebras and query languages

In Section 2.5 we elaborate more on algebras specifically developed for structured retrieval and how they can be adapted to IR, but first we present the NEXI query language and two algebras used for ranked information retrieval in structured documents (XML).

NEXI – Narrowed Extended XPath I

To overcome the drawbacks of XPath and XQuery data models with respect to IR, several data models and query languages are proposed for IR-like search over XML documents [7, 72, 209]. As the NEXI query language is used for illustrative purposes throughout the thesis and also as an end-user query language in our prototype system (TIJAH), it is briefly described here.

As the name suggests, NEXI is derived from XPath. It is simplified such that it only supports two types of nodes: element nodes and attribute nodes. It does not define character strings but terms in a document. The query language accepts only two axis steps: the *self* step (denoted with '`.`') and the *descendant* step (denoted with `//`). Also node tests for element nodes and attributes, and only value comparison ($\{>, \geq, <, \leq, =\}$) in the predicate are allowed. The extension is done in the predicate part where the specification of the *about* expression is allowed. The *about* has the form of *about*(*path*, *query_terms*). The *path* is a sequence of XPath descendant steps, starting from the self node (denoted with '.'). The *query_terms* part contains a set of terms (words or phrases), with optional modifiers ('+' or '−'), that define the terms on which the search should be performed for elements on the *path*. It is also allowed to have more than one *about* statement in one predicate combined in AND and OR expressions.

For example, the search for an `image` or a `video` depicting a 'garden' with 'flowers' in a `story` that contains `paragraph`s talking about 'beautiful flowers' and 'children' can be expressed as (example repeated from the Introduction):

```
//story[about(.//paragraph, ‘‘beautiful flowers’’ children)]
   //(image|video)[about(., flowers garden)]
```

Path algebra

Path algebra [71, 72] is developed as a part of the HyREX retrieval system [3] by Fuhr et al. HyREX follows a three-level database architecture. It uses the XIRQL XML query language for query formulation. Furthermore, the HyGate [73] user interface is developed, and it enables the formulation of user queries in XIRQL and presents the results to the user based on path algebra expression evaluation. At the physical level data is stored in a modified inverse list structure that utilizes B*-trees for speeding-up the implementation of path algebra operations.

At the logical level the XIRQL query is transformed into a path algebra expression. As the name suggests, the algebra specifies path manipulation, where a path is defined as a sequence of elements that have parent-child relationship in an XML structure. Since the XIRQL syntax is based on XQL syntax [183], one of the predecessors of XPath [37] and XQuery [18], the path algebra very much resembles the XPath/XQuery algebra. However, it operates on a set of paths instead of node-sets. The main feature of the algebra (and the system) is that it supports search term and document term weighting, as well as specificity-oriented search, i.e., retrieving the most specific parts of a document that are relevant to a query (see Section 2.1.2). Handling of IR queries is based on datatypes with vague predicates [75] and the algebra uses structured vagueness to find close matchings for structured conditions.

The authors define a set of *atomic units* for each XML document which can be search and retrieval units. These units are elements for which the term statistics are computed. Atomic units can be useful when structured conditions are not specified in the query (content-only queries) but restricts the searcher from searching in arbitrary elements in a document structure. Furthermore, atomic units have to be selected by a system administrator before the construction of a database.

Another issue addressed in the path algebra approach is handling of nesting of atomic units. The nesting might imply that the terms are counted more than once if evidence from nested units is combined. To resolve this problem authors adapt the probabilistic relational algebra approach [69, 75], where disjunctive normal form of event expression is used to model uncertain inference. Authors introduce *augmentation event* that defines the event when the atomic units are accessed from their ancestors. However, the augmentation weights have to be predefined for each pair of nested atomic units.

The structured vagueness is expressed in the form of (1) dropping the distinction between attributes and elements and representing them as vague data types, (2) similarity of element names, and (3) generalization of child/parent to descendant/ancestor relationship. The authors also point out that other types of vague data types can be used to model different type of vagueness in structured search [72].

Full match algebra

Unlike the previously presented document-centric algebra for specificity-oriented ranked retrieval, recently a data-centric algebra for full-text search extension to XQuery was presented [6, 7] (the document- and data-centric classification is given by Fuhr in [72]). The algebra is based on XML full-text search requirements [28] and use-cases [8] (later refined in [9]). The development of the algebra resulted in the reference implementation of the XQuery 1.0 and XPath 2.0 full-text extension (XQueryFT) – GalaTex [52].

The XQueryFT algebra, also called *full match* algebra, is based on a data model defined over linguistic units. A linguistic unit, or a position, consists of a token, its identifier (assigned using pre-order XML document traversal), parent XML node, relative position in a sentence, relative position in a paragraph, and context (tag name, attribute, etc.). The full match algebra is based upon a first-order logic disjunctive normal form predicates, specified over XML linguistic units.

The queries are expressed in TexQuery, a query language that is an extension of XQuery 1.0 and XPath 2.0 query languages. It provides a set of full-text search primitives called *FTSelections*. An XQuery/XPath query is evaluated on a sequence of items, while the TexQuery part is transformed into *FTSelections* that are evaluated over the full match data model. The connection between two languages and data models is established using two types of expressions: *FTContainsExpr* and *FTScoreExpr*. While the former returns only *true* or *false* with respect to the matching performed in the full match data model, the latter returns a *score* in the range $[0, 1]$ that represents 'how good' is the matching.

The specification of the retrieval model that delivers the score is not part of the full match model and is left to the implementer. The interpretation of the results of full text search using *FTContainsExpr* and *FTScoreExpr* expressions is left to the end user when formulating a query. Furthermore, once the TexQuery *FTScoreExpr* query expression is evaluated and results translated into XQuery/XPath data model the system cannot reason any more about the semantics of these scores.

By separating the DB-like XQuery/XPath data model and IR-like full match data model, authors fulfill their design goals such as full composability of *FTSelections*, powerful TexQuery language syntax, and extensibility with new primitives (see [7]). However, this was a step back for the integration of IR and DBs as the algebra completely disregards the IR process when specifying algebra operators, and therefore disables the reasoning about the retrieval models inside the algebra.

2.5 Region Algebras

This section starts with the specification of region algebra terminology and characteristics of region algebra approaches. It continues with discussing features and drawbacks of various region algebra approaches. The comparison of these region algebra approaches and recommendations for the extension of region algebra approaches for ranked retrieval are given afterward.

The data model in most of the region algebra approaches is defined as a set of regions that are originally called text extents in [27]. Regions are specified by starting and end position pairs for each region, denoted as s and e, where $e \geq s$. Region sets are organized in different concepts, such as concordance list [27] or generalized concordance list [38], which bring some constraints on how these sets can be formed or on what can be the result of the application of region algebra operators to operands (see below).

The application of the idea of text regions to structured documents is straightforward. If we regard each document instance as a linearized string or a set of *tokens* (including the document text itself), each component can then be considered as a text region.

The basic set of operators in region algebra approaches consists of containment operators – *containing* and *contained by*, containment negation – *not containing* and *not contained by*, and set operators – *union* and *intersection*. In different approaches the basic operator set is extended with special purpose operators.

Depending on the data model in some of the approaches nesting and/or overlapping of regions is permitted in the result region set and in some not. We use this classification to first present algebras that do not support nesting and then algebras that support it.

Afterward, we give two approaches for extending region algebras for ranked retrieval. Although they support ranked retrieval, ranking (scoring) is not part of the data model and is not supported in region algebra operators. The need for ranked retrieval in region algebras is illustrated in [12]:

"... a structured text retrieval system searches for all the documents which satisfy the query. Thus, there is no notion of relevance attached to the retrieval task. ... a structured text retrieval algorithm can be seen as an information retrieval algorithm for which the issue of appropriate ranking is not well established. In fact, this is an actual, interesting, and open research problem."

2.5.1 Region algebras that do not permit nesting

This section describes region algebras where operands, i.e., region sets, cannot contain nested regions and where the application of a region algebra operator to operands is always a region set that does not contain nested regions.

PAT algebra

The earliest work on region algebras, called PAT algebra, is presented by Gonnet and Tompa [78] and Salminen and Tompa [188]. The main characteristic of PAT is that the data model is defined on *character strings*, where each character is an elementary unit (match point), and *indexed strings*, where each token bounded with delimiters is considered to be an elementary unit (region). The result of the evaluation of any PAT expression can be either a *match point set* (a character position or a pattern starting position in a string) or a *region set* defined by a starting and end match points. Due to the distinction between match points and regions in the data model, expressions can be formed in PAT that produce unexpected results, especially if the first operand is a region set and the second one is a match point set. For example, the union of two region sets containing overlapping regions is a match point set [188]. Thus, in the later region algebra approaches the distinction between match points and regions has never been incorporated into the data model.

Among a number of operators PAT includes operators for expressing containment relations between regions: including (i.e., containing), not including, within (i.e., contained by), as well as set operators for forming set union, set difference, and set intersection.

Concordance lists algebra

Burkowski [27] defined a region algebra based on a text collection represented as a finite sequence of words. He identified three basic entities: words, text elements, and contiguous extents. A *text element* is defined as a sequence of *contiguous words* that have a semantic meaning (e.g., identified with markup). Each sub sequence of consecutive words represents a *contiguous extent* defined by the position of the starting word in the sequence and the position of the end word in the sequence. A data model is defined on a set of contiguous extents, used to denote the positions of words and text elements, specified using markup or another kind of document annotation.

In the algebra, two contiguous extents can be either nested or disjoint, i.e., overlapping regions are not allowed. Burkowski's model also introduces a *concordance list* as being a named list[9] of starting and end position pairs that specify disjoint contiguous extents.

Algebraic expressions are specified using the so-called retrieval command string (RCS). RCS is defined as a sequence of filter operations that specify which contiguous extents should be selected or rejected based upon containment tests: select narrow (contained by) – SN, select wide (containing) – SW, reject narrow – RN, and reject wide – RW. RCS operations are performed on concordance lists, where the left operand represents a concordance list which is filtered by a set of concordance lists that represent the right operand. For example, a search on "sections" that contain terms "beautiful" or "flowers" can be expressed as: `SECTION SW {'beautiful','flowers'}`. Therefore, in Burkowski's model the interpretation of ',' in the query is a Boolean OR (concordance list union) operator in the algebra. The author does not introduce a Boolean AND (concordance list intersection) operator as the search on "sections" that contain terms "beautiful" and "flowers" can be expressed like: `SECTION SW {'beautiful'} SW {'flowers'}`.

Furthermore, RCS syntax contains some additional operators for supporting ranked retrieval, such as cardinality, sublist, and length operators. Relevance ranking is done on-the-fly and is based on a type of *tf.idf* model (for details see [27]). Additionally, a ranking vector was introduced to store the ranking values for contiguous extents in a concordance list. However, the binding between those vectors and contiguous extents is not clearly defined in the paper. The introduced ranking operators, as well as the ranking vector definition, violates the contiguous extent data model as the result of the mentioned operators (stored in a ranking vector) is a real or natural number and not a contiguous extent.

[9]Although the author uses the "names" for the concordance lists, those names are not identified as a part of the data model. They are used only for fetching the proper concordance list (set of contiguous extents).

GC-lists algebra

To enable expressions on overlapped contiguous extents, as well as Boolean operators and operators for proximity search, Clarke et al. [38, 39] loosen some constraints of Burkowski's model. In the data model a distinction is made between content text (alphabet) and markup (alphabet). The content text bounds are in the domain of positive natural numbers, and each number represents relative position of the word in a flat document representation where markup is excluded (preserving their relative order from the original document). The markup regions are in the domain of positive rational numbers. The number is assigned in such a way that the markup symbols, denoting the beginning and the end of a structured element that begins and ends on the same word from the text alphabet, can be clearly identified. For example, ⟨speaker⟩witch⟨/speaker⟩ is indexed as: `<speaker>`^{n}, `witch`^{n}, `</speaker>`$^{n+1/2}$. For more details see [38].

The algebra is defined on a set of regions called *generalized concordance list* (GC-list). A GC-list allows only non-nested regions, while the overlapping of regions in a GC-list is supported. The algebra preserves the four basic operators from the original model of Burkowski, which are in the paper termed contained in (i.e., contained by), containing, not contained in, and not containing. Authors explicitly introduce two combination operators, "both of"and "one of", similar to Boolean AND and OR operators, and an ordering operator "followed by". The result GC-list of the application of "one of" operator is a set of non-nested extents that contain either extents from the first or extents from the second operand. The result GC-list of the application of "both of" operator is a set of non-nested extents that contain at least one extent from both operands, i.e., GC-lists. The "followed by" operator is defined as a concatenation of the closest extents in two operands.

Although the model is similar to the model presented in [27], there is an elementary difference between them. The difference is in modeling of structured data. In Burkowski's approach the initial data set consists of (1) regions in which the starting position is different from the end position (text elements) and (2) regions where those two positions are the same (words). In [38, 39] only the latter is supported. This decision made the introduction of ordering operator a necessity, as this was the only way to define the regions where starting and end positions are different. Furthermore, this made the creation of arbitrary regions possible in the algebra. Due to a well established data model and operator set, and clear specification of operator properties [38], the GC-list algebra is used as a foundation of Multitext retrieval system explained in Section 2.5.3.

2.5.2 Region algebras that permit nesting

In the region algebra approaches discussed below nesting of regions in the result region sets, as well as in the operands is permitted.

Proximal nodes algebra

A proximal node region algebra was introduced by Navarro and Baeza-Yates [150]. The algebra is defined on a set of nodes that represents either symbols (i.e. words or characters) or structured elements organized in a set of independent hierarchies. The data model is formed of a *view* (tree structure of a document), specified using a set of *constructors* (node types in the tree structure) and associated *segments* for each constructor (a pair of numbers representing contiguous portion of underlying text). Additionally, the data model has a specific *text view* composed of text constructors that have a flat structure.

The algebra supports three types of operators: (1) text pattern-matching operators, (2) operators for the selection of structured components, and (3) operators that combine other results. In the third type of operators an extensive set of inclusion (containment) operators, distance operators, and set manipulation operators is defined. The inclusion operators contain also: direct inclusion (parent-child inclusion) and positional inclusion (inclusion of proximal regions). The authors show that proximal nodes algebra is among the most expressive ones that can still be efficiently implemented [150]. Although the model is very expressive, it cannot support overlap in the result set, as well as the combination of nodes from different views.

The proximal nodes data model is later extended to support search in XML documents, based on XQL [11] and XPath [151] query languages. The extensions are mainly for handling different node types, e.g., attributes and hyperlinks, in XML, and for the preservation of numbering as defined in the original data model. In the extended framework XQL/XPath expressions are translated into the proximal nodes algebra before the evaluation.

Consens/Milo region algebra

Consens and Milo [41] studied characteristics of region algebra approaches in the scope of capturing the important structured properties of structured text documents (i.e. nesting and ordering) and investigated the complexity in the query optimization process. They used modified PAT algebra [188] in their approach, and simplified it by keeping only the core functionality of the algebra, to be able to compare it with the approaches previously described in this section.

Consens and Milo created the relationship between the region algebra and the first order monadic theory of finite binary trees which they used for proving some of the properties of the algebra. The authors especially outlined two properties:

- region algebra approaches are incapable of expressing direct inclusion for nested regions, and
- region algebra approaches are incapable of expressing both included expression, i.e., the containment relation between a region, and two regions which have to be in a specified order.

As these two properties are useful for many applications, the authors show that by embedding the algebra in a programming language with a 'while' construct and assignments, they can be supported.

Nested region algebra

The application of region algebra for search in nested text regions was presented by Jaakkola and Kilpelainen [101, 102]. The data model is defined on a set of regions where each region represents a contiguous non-empty interval of positions. Authors introduce an indexing function $I(p)$ that uses regular expressions to return a region set consisting of all the regions that bound the text pattern p occurring in a queried text. The numbering is formed on a character basis, instead of a word basis.

Operators are defined based on conditions of relative containment and ordering (start and end position orders) of regions. In the operator set all containment operators from the model presented in [38] are preserved. Furthermore, additional operators are introduced for expressing set difference (Boolean NOT operator) and for coping with nested properties of the data model: (1) binary operator *quote* for producing disjoint "followed by" regions, (2) unary operator *hull* for producing minimal set of regions that covers (contains) the regions in the original region set, and (3) binary operator *extracting* for removing nested regions.

Jaakkola and Kilpelainen showed that the nested region algebra queries can be evaluated in practice in linear time in the length of the indexed text. They have proved that their nested region algebra is capable of expressing "both included" relation (although the expression is somewhat complex [102]). The "direct inclusion" relation, however, cannot be expressed in their algebra.

Text constraints

Another region algebra approach for arbitrary text regions is described in [148, 149]. Miller et al. based their algebra on the language for specifying text structure, called text constraints (TC). Similar to a region definition in [102], authors defined the data model on a character basis. The searching is performed using three different primitive expressions: (1) literals for matching all occurrences of a string in a data model, (2) regular expressions for matching regions, and (3) identifiers that refer to predefined named regions (e.g., *tag* in HTML).

The operator specification is based on a subset of Allen's 13 interval relations [4], namely before, after, in, contains, overlaps-start, overlaps-end[10]. The operator set consists of the same operators as defined in [102], except the hull and extracting operators. The algebra also introduces new operators, such as concatenation operator which is used for defining adjacent regions. Furthermore, Miller [148]

[10]Miller has shown that the set of relations is complete and that the other seven possible relations can be defined as a union or intersection of the basic six relations.

showed that many more operators can be defined using the *iterator* operator in combination with the basic region operators.

2.5.3 Region algebras developed for ranked retrieval

The following two region algebra approaches discuss the potential application of region algebra approaches to ranked retrieval.

Region algebra with ranking

An attempt for adapting region algebra to ranked retrieval was presented in [130, 131]. The algebra is based on the region algebra presented in [38] (i.e., nesting is prohibited for result regions). However, the approach cannot be considered as a fully algebraic approach since the relevance ranking is not defined in the scope of algebraic operators. It is a side effect of the application of algebraic operators, i.e., it results in a change in values of relevance scores that are not part of the algebra data model (similarly to approach presented in [27]). In this approach, query is decomposed into a series of subqueries, each representing a subtree of a query tree. Each subquery depicts the region algebra operation evaluated on two region sets that are obtained after the execution of child subqueries in a query tree. The algebra operators are defined as in [38].

To enable ranked retrieval Masuda et al. [130, 131] extended the definition of operators in a way that each operator produces scores for the result regions as a side effect. For score computation they considered each token to be a keyword, and treated it like keywords in traditional IR systems. Using a traditional IR *tf.idf* approach authors applied term frequency and inverse document frequency computations for tokens (i.e. keywords) obtained as a result of the application of a subquery. The computed scores are used to define a *document vector* and a *query vector*. The document vector represents the vector of term frequencies computed for the results of each subquery, while the query vector represents inverse document frequency kind of measure for each subquery (for details see [131]). The final score is computed as a cosine measure between the document vector and the query vector.

In order to incorporate structured information in the retrieval model authors proposed several mappings (i.e., coefficients) that can be used to define the query vector. Besides the simple *idf* measure, two other coefficients are proposed. The *structure coefficient* takes into account the rareness of the structures. The *interpolated coefficient* is the *idf* score of the query itself, combined with the weighted average *idf* scores of its subqueries.

Multitext algebra

The Multitext [44, 45] data model is based on a model presented in [38]. However, the authors dropped the distinction between the indexing of markup and content text. As a query language authors use the GCL (GC-list) query language [39]. The retrieval units are *shortest fragments* (shortest substrings) in a document that satisfy a query. The shortest fragment denotes the substring that does not contain other substring that satisfies a query.

The relevance ranking method is termed cover density ranking (CDR) [45]. It is based on two principles: (1) an element that contains more search terms is more likely to be relevant than that containing fewer, and (2) elements that contain the same number of terms in a shorter fragment are consider more relevant (shortest substring ranking principle). The CDR method first generates tier queries, where the first tier contain all the query terms, the second tier contains all but one query term, etc., and then combine them for the final result. The retrieval model is an ad hoc model developed based on the the two mentioned principles (see [45] for more details), and takes into account the sizes of relevant fragments in the desired element.

Although the authors extend the GC-list algebra for ranked retrieval, this extension is, as in the previous approach, not algebraic. A region score is not part of the Multitext algebra data model, and region (element) scoring is done outside the algebra operators.

2.5.4 Comparison of region algebra approaches

Here we emphasize several differences in region algebra approaches. Although region algebra approaches share the same nucleus, as we already saw, region algebra can be formalized differently considering the data model and algebra operators. Table 2.4 presents a comparison of different region algebra approaches considering the features of region algebras discussed below. A more detailed comparison of region algebra approaches, with the emphasis on their expressiveness and complexity, can be found in [10].

The first distinction is based on the content modeling in the region algebra data model (*model base* column in Table 2.4). One way is to view document text as a character string and index each character in a string using its relative position with respect to the beginning of a document. Although this approach is beneficial for regular expression and substring search, it might yield many problems with respect to large document collections, if we consider that each data unit and its *id* (position) are a part of a database data model. In that case each character, possibly with its index (*id*), should be stored in a database, resulting in an enormous table (or tables) on which the manipulation should be performed. The other solution would be to index characters dynamically in a database, using the existing database operators, which would result in a very inefficient query execution.

In many approaches the text is viewed as an indexed string (the name is taken from [188]), where string consists of indexed elements (i.e. tokens or words) and delimiters (e.g., white space, line feed). Although this approach introduces some problems in distinguishing between elements and delimiters (e.g., if "." is con-

Table 2.4: Comparison of characteristics of different region algebra approaches.

Approach	**model base**	**nesting**	**overlap**	**set ops.**	**ranking**
PAT	both	no	no	all	no
concordance list	string	no	no	union	yes
GC-list	string	no	yes	no	no
proximal nodes	both	yes	no	all	no
Consens/Milo	string	yes	yes	all	no
nested regions	character	yes	yes	all	no
text constraints	character	yes	yes	all	no
ranked algebra	string	no	yes	no	yes
Multitext	string	no	yes	no	yes

sidered as a delimiter then the string "27.01" would be divided in two elements, "27" and "01" and search for the term "27.01" would fail), the indexed string manipulation and storage is prevalent in database systems.

The region algebra approaches also differ in the way how tagging information (e.g., markup) is treated in the indexing process. In case of character based indexing, markup regions are dynamically created (at query time) by recognizing the character sequences used to denote the beginning and the end delimiters of a tagged data sequence, e.g., "⟨" and "⟩" for the start tag in XML. To avoid the repetition of complex expressions for defining opening and closing tags Jaakkola and Kilpelainen [102] used macros that can be considered as complex operators for finding regions formed by opening and closing tags. In the indexed string approach, meta data has to be somehow distinguished from the document content. Therefore, Clarke et al. [38, 39] used real numbers to index markup and Burkowski [27] used "naming" for concordance lists that is predefined, i.e., defined during the database load.

Table 2.4 also points out the restrictions of different region algebra approaches with respect to the region data model on which operators can be defined. Column *nesting* specifies whether a particular region algebra approach supports operators that can produce nested regions in the result region set, while *overlap* column depicts whether region algebra approaches allow overlapping between regions. Because of some of these restrictions most of the region algebra approaches can be evaluated in linear time [102, 222].

The last two columns, *set ops.* and *ranking*, distinguish between approaches that support or do not support set operators and mechanisms for ranked retrieval respectively. Set operators defined in most algebras are union, intersection, and difference operators. Exceptions are concordance list approach that includes only set union operators (set intersection is defined implicitly through the containment operators; see Section 2.5.1) and GC-list based algebras (GC-list, ranked algebra, and Multitext algebra) that do not support any of the set operators. Among the region algebra approaches only concordance lists, ranked algebra, and Multitext algebra, support ranked retrieval. However, these region algebra approaches do not have region scores or ranked regions as part of their data models and produce relevance scores as a side effect. Following the argumentation from [12], cited at the beginning of this section, we argue that incorporating relevance ranking in region algebras is an important issue. It is one of the main topics in this thesis.

2.6 Summary

This chapter is an extensive overview of the related approaches coming from the information retrieval and database areas. It also illustrates the need to analyze structured information retrieval for defining a logical algebra for structured IR systems. It starts with the overview of traditional flat text retrieval models and explains how they can be applied to structured IR (and multimedia IR). The conclusion is that for developing structured IR models, a more systematic analysis of the structured IR task needs to be performed. Database approaches that try to integrate information retrieval with database systems are then described. The focus is on the integration using conceptual schema, i.e., logical algebra. A number of existing approaches are presented, illustrating the possibility for the integration.

Sections 2.3.2 and 2.4 explain in more detail the organization of structured documents, and present the existing query languages and algebras that are used for expressing search over structured documents. Similarities and differences between the structured document (XML) data model and relational data model are then discussed, emphasizing the need for a new algebra that would integrate DB and IR approaches on structured retrieval scenario.

The chapter ends with an overview of region algebra approaches, as they provide the basic functionality needed for the development of a new transparent algebra for structured information retrieval. The approaches are compared at the end, and the aim of the extensions of region algebras for relevance ranking (explained in detail in Chapter 3 and 4) are presented.

Chapter 3

Structured IR: Requirements & Framework

This chapter analyzes the research problem addressed in the thesis, i.e., the problem of structured information retrieval. It identifies four elementary requirements for structured information retrieval. To fulfill these requirements a mathematical framework is proposed, called *score region algebra* (*SRA*). The SRA data model and operators are discussed in detail. The chapter concludes with a discussion on variants of score region algebra and SRA features and limitations.

This chapter is partially based on papers published (1) in the Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (abstract) [137] and (2) in the Proceedings of the 14th ACM International Conference on Information Knowledge and Management (CIKM) [138].

3.1 Structured IR Requirements

The discussion on specifying structured information retrieval using various retrieval models, presented in Section 2.1.2, pointed out the need for systematic analysis of the structured retrieval problem to identify structured retrieval requirements. The basic analysis of a user request formalized as a structured query, given in Section 1.2.3, introduces the elementary structured retrieval requirements that we also follow in the development of our structured retrieval framework. These requirements are the main topic of this section. The following four elementary structured retrieval requirements are the ones that characterize structured information retrieval:

1. entity selection
2. relevance score computation
3. relevance score combination
4. relevance score propagation.

To perform detailed discussion on structured IR requirements we start with the analysis of a relatively complex user request expressed as a structured IR query. The user request is as follows:

Suppose a user recently saw a nice image of a garden with beautiful flowers in a story, that he/she would like to add to his/her collection. He/she is not sure whether the story was about children and beautiful flowers or the one that contained a nice video about gardens. To clarify the search the user also uses a sample image and a sample video clip: 'flower_garden.jpg' and 'garden.avi'.

Consider the following NEXI [209] query expressing this request:

```
//story[about(.//paragraph, children "beautiful flowers") or
      about(.//video, garden src:garden.avi)]
   //image[about(., flowers garden) and
      about(., src:flower_garden.jpg)]
```

The query states that the user searches for paragraphs in a story that are about 'children' and 'beautiful flowers' or that contain a video clip similar to a clip 'garden.avi'. He/she wants to see an image about 'flowers' and 'garden' that looks like an image 'flower_garden.jpg', inside the story.

For the analysis we first identify the basic entities that are part of a user query, illustrating them on our example query. These are: terms, search and answer elements, and multimedia content references. Our example query contains four different query terms: 'beautiful flowers', 'children', 'garden', and 'flowers'. The first one is a phrase consisting of two words, 'beautiful' and 'flowers', and the other three are single words.

We can distinguish four different structured constraints, expressed in the form of element or tag name specifications: 'story', 'paragraph', 'video', and 'image'. According to the NEXI specification, the *answer element* is the element for which an *about* predicate is specified, i.e., in our example the 'image' element. All other elements which are not answer elements are called search elements. Within the search elements, we distinguish two different kinds: the lowest element inside an about, i.e., the last element in the *about* path expression, and other elements.

There exists one special case where the lowest search element inside an about is '.', that refers to the search element that directly precedes the about clause. Thus, it can happen that the search element is the answer element at the same time (e.g., the 'image' element in our example query). In our example query, the search elements are: 'story', 'paragraph', 'video', and 'image'. The lowest search elements inside abouts are 'paragraph', 'video', and '.', i.e., 'image'.

Queries can also contain entities that are not terms or elements and that need to be addressed in the search process. For example, the following entities 'garden.avi' and 'flower_garden.jpg' are the locations (references) of the two *sample multimedia items*, namely a video clip and an image. The reference multimedia item can also have other forms than images and video, such as text files, mathematical formula, graphs, songs, etc.

3.1.1 Entity selection

The first structured IR requirement discusses the selection of different entities in the structured collection, based on the entities specified in the query. As in the query we isolated three entities (terms, search and answer elements, and multimedia content references), *entity selection* in structured IR systems must support term, element, and multimedia content selection. Furthermore, as query consists also of descendant steps (//), entities can be selected based on their containment relation (explained in Section 3.2.4).

Terms

The search request in flat text documents is usually expressed as a list of query terms. Therefore, the terms can be considered as the elementary units used for modeling search over textual documents (see Section 2.1). A term is a sequence of characters from an alphabet that has some commonly accepted semantic connotation. It is also an elementary unit of any language. Therefore, the motivation for the usage of terms as an elementary units for the development of IR models is straightforward.

The term selection requirement is about finding the location of a search term in the collection of documents to be searched. However, from the IR perspective not only elements/documents that contain that particular term are relevant. For example, in our example query paragraphs that contain the term 'child' instead of the required term 'children', or even the term 'kid', might also be considered relevant. Therefore, the term selection does not have to select only the terms that exactly match the search term, but also terms that are in some way similar to the search term. This kind of vague search can be explicitly stated in a user query or left to the system's intelligence.

In IR systems, the two most common approaches to incorporate this vague term selection are to employ *stemming* and *query expansion* by using e.g., term synonyms. Stemming is a method of reducing terms to their stem or root form. In such a way the term selection can be performed on all the terms that represent different derivation of the root word. Term synonym selection selects all the terms that have the same or nearly the same meaning as the search term. Similarly, a word whose meaning is a specific instance of a more general word – hyponym, or the other way around, a word whose meaning is a generalization of a meaning of other word – hypernym, can be used. Thus, using stemming or synonym, hyponym, hypernym, and other query expansion techniques, we can implement the vague term selection.

Search and answer elements

Similarly as selection of terms can be vague or strict, search and answer element selection can also be vague or strict. In case the user is familiar with the collection he is querying, i.e., he knows the structure of documents in the collection, he can pose a query where elements should be strictly matched. In this case only the elements that exactly match the search and answer elements specified in the query should be selected.

This is useful for the expert users that know the details of the document structure and would like to find some specific information in a part of a document. The strict search can also be helpful for the non-expert users. The prerequisite for such strict element search is that a mechanism exists for helping non-expert users to specify precise structured queries. For example, by graphically illustrating the structure of documents.

Strict element search can also convey a higher level of abstraction, where several document components identified by different element names are considered as equivalent in the retrieval process. These elements should denote the same semantic concept. For example, a list of equivalent element names is used in the INEX collection (see [209] for the complete list of equivalent names). In such a way the query specification process (for non-expert users) could be simplified and the underlying selection process would have to cope with this abstraction of strict element selection.

However, due to structured document heterogeneity (i.e., usage of different document schemas), different documents may have different element names for the same concept – 'element name synonyms'. Also, single document can have more than one structured description (see e.g., [155]). Furthermore, the user might not be certain where he would like to search for information or he might not know the element names in the collection. In all cases, the user should be able to give some hints to the system, such as where he would like to perform the search and what he would like to see as an answer.

The problem of the vague element name matching is studied in the research area of schema matching and numerous techniques exist that try to resolve this problem (see [63, 176] for a survey). However, the schema matching approaches are concerned with matching the relations among elements (including the containment relation) besides element name matching. In this thesis we only focus on a strict search and answer element selection. The vague search is beyond this thesis, but in [145] we explain how simple element name matching can be supported in our framework.

Multimedia content

The selection of multimedia entities by matching them with the sample multimedia item (referenced in the query) is a problem studied in the multimedia retrieval area. The goal is to select multimedia data that corresponds to the sample multimedia data. For example, an image similar to another (sample) image, or a song similar to another (sample) song. Although the search can be strict, i.e., find an image, video, audio, or even a text document, that are exactly the same as the reference one, the vague multimedia content search (selection) is much more probable scenario.

While for the terms and elements vague selection could be modeled simply by choosing term synonyms or 'element name synonyms', the concept of multimedia content synonyms does not exist. Furthermore, in case of non-textual documents, multimedia bears greater complexity than simple words. To be able to select multimedia content based on a similarity criterion, multimedia data is described in terms of low-level or high-level multimedia features, as described in Section 2.1.3. Therefore, for such multimedia content selection we talk about multimedia feature matching instead of term or element name matching.

Multimedia feature matching can be classified based on the type of multimedia content described by features, e.g., image features, video features, audio features. To be matched, features usually form an input to an 'intelligent' matching framework, such as Bayesian net [143] or Gaussian mixture model [217]. The output of the matching is a value that depicts the degree of similarity between a sample multimedia content and the multimedia content that is contained in a searched document. To restrict the search, a threshold can be applied to filter non-relevant multimedia content. Thus, only the multimedia content that is similar to the sample multimedia content is selected. This is the approach that is also followed in this thesis (in Chapter 7) for the multimedia content selection. As multimedia retrieval models are not the topic of this thesis, we do not go into details how the matching among multimedia data is performed.

3.1.2 Relevance score computation

After the selection, the relations between these selected entities should be established. In the example NEXI query, the first relations are the ones in the *about* statements, i.e., paragraphs should be about 'beautiful flowers' and about 'children', videos should be about 'garden' and look like 'garden.avi', and images should be about 'flowers' and 'garden' and look like 'flower_garden.jpg'. This step corresponds to determining how relevant are elements 'paragraph', 'image', and 'video' to the user request, with respect to terms or multimedia content they contain. Thus, we termed this requirement element *relevance score computation*.

The relevance score computation requirement, based on the type of search modeled with it, consists of several aspects. For example, the relevance of an element can be computed with respect to its textual, video, or image content. In the example query, relevance scores have to be determined for the 'paragraph' elements with respect to two terms: 'beautiful flowers' and 'children'. For the 'video' element we have to compute scores with respect to term 'garden' and the containing video clip similar to the 'garden.avi'. In the other predicate, the scores for the 'image' element need to be determined with respect to terms 'flowers' and 'garden' and reference image "flower_garden.jpg'. Therefore, different classes of structured element relevance score computation can be distinguished. We distinguish between two major classes: element-term score computation, and query-by-example score computation.

Element-term relevance score computation

Element-term score computation can be considered as a variant of the document relevance score computation in the flat text retrieval. The difference is that in document component retrieval, the relevance score is computed for a term in an arbitrary structured element, instead of a predefined document. The distribution of terms in structured elements is quite different than in documents. Furthermore, elements greatly vary in size; some elements contain only a singly word, while others contain more than hundred words. The question is how such different element granularity and different distribution of terms inside elements can be incorporated into the element-term relevance score computation.

In flat text IR systems, the relevance score of a document is usually based on a number of occurrences of a term in a document (term frequency). Possibly the additional statistical information, like collection frequency or inverse document frequency of a term is used, as explained in Section 2.1. Thus, the element relevance score computation should be also based on a number of occurrences of a term in an element. Additionally, the scoring mechanism can include information that depicts element-term statistics, such as inverse element frequency, but also the statistical information of other contained or containing elements in a structured hierarchy. Several approaches for element-term relevance score computation are explained in the next chapter.

Some query languages (IR systems) enable the distinction between terms that are more important to the user than others (denoted with, e.g., '+') and the specification of terms that are not wanted in the answer element (denoted with, e.g., '−'). However, how '+' and '−' are modeled in the retrieval process is left to the retrieval system implementer. '+' might mean that the search term must be present, while '−' that the search term must not be present. Equally likely, '+' might also mean that the term is more important than other terms in the search process or e.g., that it should not be stemmed in the search process. This should be reflected in the element-term relevance score computation.

Some retrieval models also include phrase search and proximity search. For phrase search, the relevance score computation should somehow reward elements that contain the search phrase in contrast to elements that contain single words but do not contain the phrase [106, 187, 225]. Similarly, elements that contain terms close to each other should be rewarded in proximity search [45, 150]. However, term modifiers, phrase, and proximity search are not discussed in this thesis (see [144] for more details).

Query-by-example score computation

For the textual content search, element relevance score computation describes the transformation from element, document, and collection term statistics to element score. However, the query-by-example score computation requirement is merely about the aggregation of scores depicting the similarity between the sample multimedia item and the multimedia items from the collection. This is due to the fact that in, e.g., the image selection process, the relevance score is assigned to images with respect to their similarity to the sample image. Therefore, the query-by-example score computation part only needs to describe how matching scores of images in the collection should be aggregated to produce the relevance score of a structured element that contains these images. While this aggregation can have similar form as element-term relevance score computation, it can also be a simple summation of scores of contained images, or their average, maximum, minimum, etc.

The query-by-example score computation scenario is the same for content types other than images, such as videos, songs, text documents, etc. The matching scores obtained after multimedia content selection are aggregated and assigned to the containing element in which the user searches for a multimedia item. However, the prerequisite for the computations of query-by-example region scores is that the scores are normalized such that they can be later combined with scores obtained when using other score computations, i.e., element-term relevance score computation.

3.1.3 Relevance score combination

Structured element search is usually performed with respect to more than one term (or multimedia content) in an element, as can also be seen in our example query. As for the score computation requirement, scores are computed on a per term (or single multimedia content) basis, those scores have to be combined. This requirement is called *score combination.* Besides the implicit score combination among elements containing different terms (e.g., paragraphs containing the term ‘children’ and paragraphs containing the term ‘beautiful flower’ in our example query), structured search allows explicit specification of AND and OR combination among elements that contain terms or multimedia content (e.g., the combination of *about* clauses in NEXI, as can be seen in our example query). To distinguish between the two, we call the former low-level score combination and the latter high-level score combination.

The score combination aspect can take two forms: either Boolean OR-like score combination or Boolean AND-like score combination, as can also be seen in the example query. Furthermore, the combination can be specified among different elements in the document hierarchy, such as among ‘paragraph’ and ‘video’ elements in the first predicate of our example query. Furthermore, it can be among elements that contain different content, e.g., among ‘image’ elements that contain terms 'flowers' and 'garden' and 'image' elements that contain images similar to 'flower_garden.jpg'.

Low-level score combination

The combination of relevance scores among elements that are directly scanned for their content information is called *low-level* score combination. For example, this is the case for the lowest search element in the *about* clause in the NEXI query specification. Although NEXI does not allow explicit specification of combination type (OR or AND) among terms, it is assumed in many structured retrieval systems that it is an AND-like combination [74]. Such combination resembles the score combination in document retrieval, however, here it is done at the element level. In other words, elements that are combined are usually smaller, and their relevance scores might be different than document relevance scores.

Such low level score combination might include score combination among elements obtained when searching different content information inside them. In our query example, search in 'video' elements for term 'garden' is combined with the search for the same elements with respect to the video they contain that need to be similar to 'garden.avi'. To be able to combine these scores they have to be normalized as stated before.

High-level score combination

High-level score combination is performed among elements that are not directly tested on their content. This is the combination of element scores where at least one element contains or is contained by other elements (that contain other elements or some content). This can be seen in the first predicates of our query expression. Although it might seem that the combination should be performed among 'paragraph' and 'video' elements, the combination should actually be performed among 'story' elements containing such 'paragraph' and 'video' elements. Even though in case the search element in the second about clause would be the 'paragraph' element, still the combination should be performed on 'story' elements. In this case the user would be searching for 'story' elements that contain either a paragraph about 'children' and 'beautiful flowers' or a paragraph about garden or video clip 'garden.avi', where these two paragraphs need not be the same.

These examples also illustrate the need for propagating scores from containing ('paragraph') to contained ('story') elements. The score propagation requirement is explained below.

3.1.4 Relevance score propagation

To be able to perform high-level score combination, the scores of containing or contained elements need to be propagated to the common ancestor or descendant element respectively. This requirement is called *score propagation.* It follows the semantics of the NEXI path expression that expresses an ancestor-descendant relationship. In case of consecutive descendant search elements, the ancestor search element can have multiple matching descendant search elements to which scores have to be propagated. Also descendant elements might have multiple ancestor elements to which scores have to be propagated.

Score propagation aspect is recognized by Fuhr et al. [71] and Grabs and Schek [82]. In their paper it expresses propagation among parent and child elements in a hierarchical structure. We can define such element score propagation as the translation of scores to the parent/child or ancestor/descendant elements where these scores can be combined based on the type of combination explicitly specified in the query language (e.g., NEXI). We can distinguish two types of score propagation: *upwards* and *downwards* score propagation[1].

Upwards score propagation

The necessity of the propagation to the common ancestor element can be seen in case of the first predicate in our example NEXI query. To be able to combine scores of paragraphs about 'beautiful flowers' and 'children', and videos about 'garden' and containing 'video' elements similar to 'garden.avi', we need to propagate scores to the common ancestor 'story' element. This propagation is not trivial as usually one element can have more than one descendant elements with the same element name which can also be nested. Therefore, the scores of many descendant elements must be taken into account when determining the score of an ancestor element.

Although specifying child/parent upwards (and also downwards) score propagation would give more control over the propagation process it would also make the propagation more complex (see [155, 170]). This is why we focus on the ancestor-descendant relationships.

Downwards score propagation

In our example query the scores of the 'story' element need to be propagated to the contained 'image' element for determining the relevance of an answer element ('image'). The relevance of the 'image' element with respect to its content, i.e., terms 'flowers' and 'garden' and image 'flower_garden.jpg' is not the relevance of the complete query but only the relevance with respect to the second predicate. To determine the right relevance score, the relevance score of the ancestor 'story' element must be propagated to probably more than one 'image' element. How this propagation is performed is specified in the downwards score propagation aspect.

Although it is not frequently the case, one descendant element ('image') can have more than one ancestor element from which the scores need to be propagated. For example, this would happen if we would have nested 'story' elements.

[1]Note that other types of score propagation can be defined, such as following, preceding, anchor, or link score propagation. However, we focus on ancestor-descendant score propagation as it is the most frequently exploited feature in structured retrieval.

Therefore, downwards score propagation also has to take care of multiple (nested) ancestor elements, similarly as upwards score propagation has to take care of multiple descendant elements.

3.2 Score region algebra

We start this section by illustrating the need for the new algebra specifying structured information retrieval, following the elementary structured IR requirements. Then we explain how we extend the basic region algebra approach to model components of structured documents and to express structured retrieval. We end the section by explaining why we use this particular score region algebra data model and operator set.

3.2.1 Why do we need an algebra?

The question is do we really need a new data model and operator set for specifying structured document retrieval, or we can utilize data models and operators modeling similar problems. The relational model would seem like a reasonable choice as shown in Section 2.4.

The hierarchically structured (XML) data model can be easily transformed into a relational one. The relational algebra operators can express containment operators (see Equations 2.28 and 2.29 in Section 2.4). However, relevance ranking cannot be easily expressed using relational algebra operators, as explained in Section 2.4. To be able to use the information depicting the relevance of an element, an additional attribute for each relational (region) table has to be introduced, as shown in [72]. The attribute should keep the ranking score values for particular structured element during the query execution. Furthermore, to avoid the repetition of relational algebra expressions in defining IR expressions, new operators should be (re)defined. Their combination should express the scoring (relevance ranking) mechanism.

Many issues exist that are not in favor of using relation algebra for ranked retrieval. Among a number of reasons to define a new algebra for structured IR, we argue that the most important ones are:

1. relational algebra cannot provide information retrieval model independence and content description independence
2. relational expressions for structured IR are highly dependent on a relational schema used for modeling structured data
3. relational algebra is not suitable for expressing IR like operations, as well as supporting their execution at the physical level.

The most important reason for defining an algebra is to enable the specification of operators that can model structured information retrieval queries (e.g., *about* in NEXI), i.e., scoring and ranking of XML elements, but abstracting away from the retrieval model implementation at the same time. Such algebra would provide *information retrieval independence* and *content description independence.* It would enable the (transparent) implementation of various retrieval models, independently of the application at the end-user level, and without affecting the algebra data model and operator definitions. Also, the algebra would not be influenced by the document content representation at the physical level, and how this document content is used for specifying retrieval models. Furthermore, the reasoning that can be done at the logical level, i.e., on the algebra expressions, can be useful for query rewriting and *optimization.* Using knowledge about the size of the operands and the cost for the execution of different operators at the physical level, we are able to generate different logical query plans achieving faster execution times and lower usage of main memory when executing at the physical level.

Another important issue concerning immediate translation of structured IR expressions into relational algebra is that the algebraic expressions are highly dependent on the relational schema chosen for the representation of structured data. If we change the relational schema, the relational algebra expressions for each query has to be rewritten according to the relational schema. This is especially the case for XML, since usually huge relational tables, which have more than a million entries, are typically broken into a number of smaller ones using one of the fragmentation methods explained in Section 2.4.1. Although this can be circumvented using views over relational tables, it would still require managing these views and using complex relational expressions for implementing ranked retrieval underneath.

The last point emphasize that to express structured IR (e.g., NEXI) queries in relational databases we need new operators for defining structured IR subexpressions, such as descendant and ancestor steps, containment conditions, etc. These new operators, however, would not follow the main ideas of the relational model and relational operators. Additionally, the exact technique how we can implement the subexpression is defined at the physical level, and it does not have to be unique, i.e., we can have multiple variants of a physical expression for the same structured IR relational subexpression. The execution times for distinct implementations can differ regarding the relational storage of the structured data, parameters of storage structures, and index structures used for the acceleration of relational expression execution in relational databases. Therefore, special techniques need to be developed for handling such expressions at the physical level that are beyond the physical operators in relational database systems.

These reasons force us to use a different algebra for specifying structured IR and not use the relational one. Our choice is region algebra [27, 38, 101, 150, 188]. While relational algebra is tailored to work on flat non-nested relations, region algebras are tailored to express search in structured documents, often having nested structures. The region algebra framework is a special-purpose framework that provides only the basic data model and operator set for structured search. It can easily be extended to support more richer data models and a richer operator set.

Figure 3.1: Enumerated example XML document.

<story^{0} id^{1}="78"^{2}>
  <title>^{3}The^{4} Selfish^{5} Giant^{6}</title>^{7}
  <author>^{8}Oscar^{9} Wilde^{10}</author>^{11}
  <image^{12} src^{13}="Garden.jpg"^{14}>
    <title>^{15}Rock^{16} garden^{17}</title>^{18}
  </image>^{19}
  <p>^{20}Every^{21} afternoon^{22}, as^{23} they^{24} were^{25} coming^{26} from^{27} school^{28}, the^{29} children^{30} ... </p>^{45}
  <p>^{46}It^{47} was^{48} a^{49} large^{50} lovely^{51} garden^{52}, with^{53} soft^{54} green^{55} grass^{56}. Here^{57} and^{58} ... </p>^{188}
    ...
  <p>^{1043}And^{1044} when^{1045} the^{1046} children^{1047} ran^{1048} in^{1049} that^{1050} afternoon^{1051}, they^{1052} found^{1053}
    the^{1054} Giant^{1055} lying^{1056} dead^{1057} under^{1058} the^{1059} tree^{1060}, all^{1061} covered^{1062} with^{1063} white^{1064}
    blossoms^{1065}.</p> ^{1066}
</story>^{1067}

Additionally, it provides a simple framework for manipulation on nested regions that can be extended to handle scoring mechanisms following the identified four elementary structured retrieval requirements. Thus, the introduction of scoring operators in region algebra is easier and more elegant than in the relational algebra, as can be seen in next sections.

3.2.2 Region algebra basics

Region algebras [27, 38, 101, 150, 188] are based on viewing documents as a set of regions instead of characters or content words and tags. Each region is usually defined by its *starting* and *end* position. Usually, some restrictions are forced on regions, such as that they cannot nest or overlap (see Section 2.5). For structured retrieval, structured documents are transformed into region sets on which a number of operators are defined. These operators test regions on their containment relations and generate new region sets by forming a union or intersection on these region sets.

The most common region algebra approach is the one where each content word in a document represents one region that has the same starting and end position. On the other hand, markup (structured) elements define a region with different starting and end positions, embracing all the words that are inside the marked up part, i.e., an element of a document.

An example XML document is given in Figure 3.1, where ordinal numbers are assigned to each token denoting its absolute position in a document. Following this view on structured documents, and allowing nesting among regions, we can define region sets for each token as given below. Here, we used words in uppercase to denote region sets for different tokens[2].

[2]For simplicity we do not distinguish between document markup and content words for now.

Table 3.1: Basic region algebra operators.

Operator	Operator definition
$R_1 \sqsupset R_2$	$\{r_1 \| r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1.s < r_2.s \wedge r_1.e > r_2.e\}$
$R_1 \sqsubset R_2$	$\{r_1 \| r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1.s > r_2.s \wedge r_1.e < r_2.e\}$
$R_1 \sqcap R_2$	$\{r \| r \in R_1 \wedge r \in R_2\}$
$R_1 \sqcup R_2$	$\{r \| r \in R_1 \vee r \in R_2\}$

$$\begin{aligned}
\textsc{story} &= \{(0,1067),...\}\\
\textsc{id} &= \{(1,2),...\}\\
78 &= \{(2,2),...\}\\
\textsc{title} &= \{(3,7),(15,18),...\}\\
\textsc{the} &= \{(4,4),...,(1046,1046),...\}\\
\textsc{garden} &= \{(17,17),(52,52),...\}\\
\textsc{p} &= \{(46,188),...,(1043,1066),...\}\\
&...
\end{aligned}$$

Table 3.1 defines the following four basic region algebra operators that can be evaluated on such region sets: containing ($\sqsupset$), contained by ($\sqsubset$), region set intersection ($\sqcap$), and region set union ($\sqcup$). We use R_i $(i = 1, 2, ...)$ to denote the region sets, r_i to denote regions in these region sets, and $r_i.s$ and $r_i.e$ to denote region start and region end positions.

As an example, in Equation 3.1 we present an SRA expression that corresponds to the following simplified query from Section 3.1 (assuming that the *about* statements require strict search):

```
//story[about(.//paragraph, children beautiful flowers) or
   about(.//video, garden)]//image[about(., flowers garden)]
```

$$\begin{aligned}
(R_p \quad :=) \quad & (\textsc{image} \sqsubset ((\textsc{story} \sqsupset (((\textsc{p} \sqsupset \textsc{children})\\
& \sqcap (\textsc{p} \sqsupset \textsc{beautiful})) \sqcap (\textsc{p} \sqsupset \textsc{flowers})))\\
& \sqcup (\textsc{story} \sqsupset (\textsc{video} \sqsupset \textsc{garden}))) \sqsupset \textsc{flowers})\\
& \sqcap \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3.1)\\
& (\textsc{image} \sqsubset ((\textsc{story} \sqsupset (((\textsc{p} \sqsupset \textsc{children})\\
& \sqcap (\textsc{p} \sqsupset \textsc{beautiful})) \sqcap (\textsc{p} \sqsupset \textsc{flowers})))\\
& \sqcup (\textsc{story} \sqsupset (\textsc{video} \sqsupset \textsc{garden}))) \sqsupset \textsc{garden})
\end{aligned}$$

In this expression operator $\sqsupset$ does not make a difference if regions in the right operand are term or element regions. Although this does not play an important role for strict search, in structured IR search this distinction is a necessity. It results in distinguishing score computation and score propagation requirements, as discussed in Section 3.1.

3.2.3 Score region algebra data model

The simple region model is not sufficient for expressing all the information in structured documents (see Section 2.3.2). The region algebra data model has to be extended to differentiate between elements, attributes, terms, and other components of structured documents, and more importantly to express region relevance scores. Also, the operators need to be defined that enable element scoring and ranked retrieval. That is why we develop a new algebra, called *score region algebra – SRA*.

In the specification of the score region algebra data model we distinguish between different entity types and names in structured documents. Furthermore, we enrich the original model with a region score attribute and introduce a number of operators for score manipulation. The aim of the SRA is to support ranked retrieval as a part of the algebra, and not as a side effect, which distinguishes it from other region algebra proposals that include ranked retrieval (see Section 2.5).

The logical data model of SRA is based on *region sets*. It is specified using the following five definitions.

Definition 1. *Region tuple r, $r = (r.s, r.e, r.n, r.t, r.p)$, is defined by these five attributes: region start attribute – s, region end attribute – e , region name attribute – n, region type attribute – t, and region score attribute – p.*

Definition 2. *Region start and region end attributes must satisfy ordering constraints: $r_i.e \geq r_i.s$.*

Definition 3. *If $\prec$ is defined as in the following equivalence $r_i \prec r_j \Leftrightarrow r_j.s < r_i.s \leq r_i.e < r_j.e$ and $\equiv$ as in the equivalence $r_i \equiv r_j \Leftrightarrow r_i.s = r_j.s \wedge r_i.e = r_j.e$, for two arbitrary region tuples it is either $r_i \prec r_j$, $r_i \equiv r_j$, $r_j \prec r_i$, or they are not nested or equal.*

Definition 4. *Region set $\mathcal{R}$ represents a set of all possible region tuples ($r \in \mathcal{R}$).*

Definition 5. *The SRA data model is defined on the domain of region power set $\mathcal{P}(\mathcal{R}) = \{\mathcal{R}' | \mathcal{R}' \subseteq \mathcal{R}\}$ which is the set of all possible subsets of the region set $\mathcal{R}$.*

The semantics of *region start* and *region end* attributes are the same as in other region algebra approaches: they denote the bounds of a region. The domain of region start and region end attributes is the domain of positive integers including 0. Region bounds are determined based on assigning ordinal numbers to tokens in a structured document (the pre-order document tree traversal) as depicted in Figure 3.1.

The *region name* attribute is used to denote the names of different entities in structured documents. The set of entities includes content words, element nodes, element attributes, element attribute values, etc. The domain of entity name region attribute is the domain of alphanumeric strings. For example, the name of the content word 'garden' is `garden`, while the name of the paragraph structured element enclosed in opening and closing markup tags, $\langle \texttt{p} \rangle$ and $\langle /\texttt{p} \rangle$, is `p`.

To distinguish between different name 'roles' in structured documents we use the region type attribute. For example, *node* is used for the element node, *text* for the text node, *word* for the word present in a text node, *attr_name* and *attr_value* for attributes, etc. The domain of region types is a set of types that are present in structured documents (and supported in SRA) and that are necessary for specifying ranked search in structured documents, i.e. $type = \{\text{node}, \text{text}, \text{word}, \text{text}, \text{attr\_name}, \text{attr\_value}\}$. The domain of region types can be easily extended in case more entity types are needed.

Finally, the region score attribute is used to specify the relevance score of a region with respect to a given query. The domain of the region score attribute is the domain of positive real numbers in the $[0, 1]$ range[3]. While all other attributes can be specified based on the structured data in the collection, a default score value for each region in the collection does not have a unique unambiguous value. It can be the same for all regions, e.g., 0 or 1, or vary based on different criteria, such as the level of the region in a hierarchical structure, its size, its estimated prior relevance to the user, etc. The choice of the default region score also depends on the retrieval model instantiation as for some choices default score of 0 or 1 would produce bad results (e.g., 0 for the models presented in the following chapter). For the sake of simplicity we assume that the default score value for regions is 1, unless stated otherwise.

Based on the SRA data model, the region sets for our example XML document (given in Figure 3.1) in score region algebra can be defined as follows. We classify regions based on the region name attribute (denoted with "region name" −− on the left side) for comparison with the basic region algebra model, although all regions are part of a large region set defining the whole data collection.

[3]The full domain of real numbers can also be used in case score values are not normalized or the logarithmic scale is used.

STORY	−−	$\{(0, 1067, \text{story}, \text{node}, 1.0), ...$
ID	−−	$(1, 2, \text{id}, \text{attr\_name}, 1.0), ...$
78	−−	$(2, 2, 78, \text{attr\_val}, 1.0), ...$
TITLE	−−	$(3, 7, \text{title}, \text{node}, 1.0), (15, 18, \text{title}, \text{node}, 1.0), ...$
THE	−−	$(4, 4, \text{the}, \text{word}, 1.0), ..., (1046, 1046, \text{the}, \text{word}, 1.0), ...$
GARDEN	−−	$(17, 17, \text{garden}, \text{word}, 1.0), (52, 52, \text{garden}, \text{word}, 1.0), ...$
P	−−	$(46, 188, \text{p}, \text{node}, 1.0), ..., (1043, 1066, \text{p}, \text{node}, 1.0),$
...	−−	$... \}$

3.2.4 Score region algebra operators

As XML is the prevalent standard for structuring documents, here we present the operator set that is tailored toward hierarchically structured documents such as XML. The score region algebra operators are explained using the example XML document given in Figure 3.1. We give the definition of only basic score region algebra operators for text search as the main focus of our research is textual search in structured documents. The extensions for more advanced text search as well as for multimedia retrieval are explained in Chapters 5 to 7. The functionality of operators is explained on the following simplified query from Section 3.1:

```
//story[about(.//paragraph, children beautiful flowers) or
   about(.//video, garden)]//image[about(., flowers garden)]
```

The basic SRA operators are defined in Table 3.2. In the specification of region algebra operators we use R_i $(i = 1, 2, ...)$ to denote the region sets, corresponding non-capitals to denote regions in these region sets (r_i), and $r_i.s$, $r_i.e$, $r_i.n$, $r_i.t$, and $r_i.p$ to denote region attributes. With C we denote the set of all regions in the working collection and with *Root* we denote the (artificial) root element for the whole collection.

The operators in SRA take one or two region sets as operands and produce a region set as result. The first three operators enable Boolean selection of regions based on their attributes or containment relations. The other six operators specify score manipulation among regions. To enable transparent instantiation of different retrieval models following the four elementary structured retrieval requirements, SRA operators are defined using (1) *abstract scoring functions* (f) that take as parameters a region from the left operand region set (r_1) and the right operand region set (R_2) and (2) *abstract operators* ($\otimes$ and $\oplus$), as explained below.

Entity selection operators

The selection operators (σ,$\sqsupset$,$\sqsubset$) have two variants. The first one (σ) selects regions by matching their description attribute values, i.e., element name and type. The

Table 3.2: Score region algebra operators.

Operator	**Operator definition**
$\sigma_{n=name,t=type}(R)$	$\{r \mid r \in R \wedge r.n = name \wedge r.t = type\}$
$R_1 \sqsupset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}$
$R_1 \sqsubset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1 \prec r_2\}$
$R_1 \sqsupset_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\sqsupset}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \not\sqsupset_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\not\sqsupset}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \sqcap_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, r_1.p \otimes r_2.p) \mid r_1 \in R_1 \wedge r_2 \in R_2$ $\wedge\ (r_1.s, r_1.e, r_1.n, r_1.t) = (r_2.s, r_2.e, r_2.n, r_2.t)\}$
$R_1 \sqcup_p R_2$	$\{(r.s, r.e, r.n, r.t, r_1.p \oplus r_2.p) \mid r \in R_1 \wedge r \in R_2$ $\wedge\ ((r.s, r.e, r.n, r.t) = (r_1.s, r_1.e, r_1.n, r_1.t)$ $\vee\ (r.s, r.e, r.n, r.t) = (r_2.s, r_2.e, r_2.n, r_2.t))\}$
$R_1 \blacktriangleright R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\blacktriangleright}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \blacktriangleleft R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\blacktriangleleft}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$

second variant ($\sqsupset$ and $\sqsubset$) exploits the containment relation among regions to perform region selection.

The first operator given in Table 3.2 ($\sigma_{n=name,t=type}(R)$) specifies the selection based on the name and type region attributes. For example, the selection of paragraph element node regions (denoted with p) from the whole collection (C) can be expressed as $\sigma_{n=\text{p},t=\text{node}}(C)$. This selects the document components bound with ⟨p⟩ and ⟨/p⟩, as depicted with a dashed bounding box in Figure 3.2. The result is the following region set:

$$\begin{aligned}&\sigma_{n=\text{p},t=\text{node}}(C) = \\ &\quad \{(20, 45, \text{p}, \text{node}, 1.0), (46, 188, \text{p}, \text{node}, 1.0), ..., (1043, 1066, \text{p}, \text{node}, 1.0)\}.\end{aligned} \tag{3.2}$$

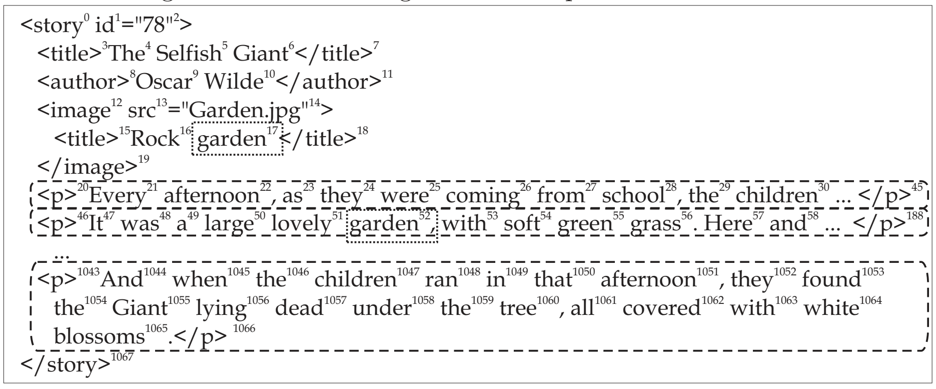
Similarly, $\sigma_{n=\text{garden},t=\text{word}}(C)$ selects all regions that actually represent the word 'garden'. Thus, the result should be the following region set (depicted with dotted bounding boxes in Figure 3.2):

$$\sigma_{n=\text{garden},t=\text{word}}(C) = \{(17, 17, \text{garden}, \text{word}, 1.0), (52, 52, \text{garden}, \text{word}, 1.0), ...\}. \tag{3.3}$$

Leaving the selection criterion for one of the attributes unspecified corresponds to a wild-card, i.e., $\sigma_{t=node}(R)$ selects all regions that have an XML element node type, regardless of their name attribute.

The other two selection operators select regions based on their containment relations, i.e., regions that contain other regions ($\sqsupset$), or regions that are contained

Figure 3.2: Selected regions on example XML document.



```
<story⁰ id¹="78"²>
  <title>³The⁴ Selfish⁵ Giant⁶</title>⁷
  <author>⁸Oscar⁹ Wilde¹⁰</author>¹¹
  <image¹² src¹³="Garden.jpg"¹⁴>
    <title>¹⁵Rock¹⁶ garden¹⁷</title>¹⁸
  </image>¹⁹
  <p>²⁰Every²¹ afternoon²², as²³ they²⁴ were²⁵ coming²⁶ from²⁷ school²⁸, the²⁹ children³⁰ ... </p>⁴⁵
  <p>⁴⁶It⁴⁷ was⁴⁸ a⁴⁹ large⁵⁰ lovely⁵¹ garden⁵², with⁵³ soft⁵⁴ green⁵⁵ grass⁵⁶. Here⁵⁷ and⁵⁸ ... </p>¹⁸⁸
    ...
  <p>¹⁰⁴³And¹⁰⁴⁴ when¹⁰⁴⁵ the¹⁰⁴⁶ children¹⁰⁴⁷ ran¹⁰⁴⁸ in¹⁰⁴⁹ that¹⁰⁵⁰ afternoon¹⁰⁵¹, they¹⁰⁵² found¹⁰⁵³
    the¹⁰⁵⁴ Giant¹⁰⁵⁵ lying¹⁰⁵⁶ dead¹⁰⁵⁷ under¹⁰⁵⁸ the¹⁰⁵⁹ tree¹⁰⁶⁰, all¹⁰⁶¹ covered¹⁰⁶² with¹⁰⁶³ white¹⁰⁶⁴
    blossoms¹⁰⁶⁵.</p> ¹⁰⁶⁶
</story>¹⁰⁶⁷
```

in other regions ($\sqsubset$). For example, the result of evaluating the following expression

$$(R_a :=)\ \sigma_{n=\text{story},t=\text{node}}(C) \sqsupset \sigma_{n=\text{p},t=\text{node}}(C) = \{(0, 1067, \text{story}, \text{node}, 1.0)\} \quad (3.4)$$

is a region that represents all the story elements that contain paragraph elements. In our case this is the example story by Oscar Wilde. Similarly, the following expression gives the set of all 'image' regions in the story:

$$\sigma_{n=\text{image},t=\text{node}}(C) \sqsubset R_a = \{(12, 19, \text{image}, \text{node}, 1.0), ...\}. \quad (3.5)$$

Although in the specification of selection operators we use strict interpretation, i.e., strict selection, other variants are possible. For example, we can use word and element name stemming (e.g., $\sigma^{stem}_{n=name,t=type}(R)$). These operators are extensions of the basic SRA operator set, and are discussed in Chapter 4 as well as in Chapter 7, along with extensions for multimedia content selection.

Element relevance score computation operators

The operators $\sqsupset_p$ and $\not\sqsupset_p$ model element relevance score computation. The relevance score computation operators assign scores to the search elements, i.e., regions from the left operand, based on attributes of regions from the right operand. Two variants of operators exist. One models the aspect where regions from the left operand should contain relevant regions from the right operand, denoted with $\sqsupset_p$. The other models the opposite, i.e., regions from the left operand should not contain regions from the right operand. The result of the application of such operators on two region sets is the region set that contains regions having the same start, end, name, and type attributes as regions in the left operand. Only the score attributes are changed, taking into account the attribute values from the regions in the left and in the right operand.

As an example, the search on paragraph elements that should contain the term 'children' (`about(.//p, children ...)`) can be expressed as follows:

$$\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{children},t=\mathrm{word}}(C) \tag{3.6}$$

In case the user would not like to see 'children' in a paragraph (i.e., in case the first NEXI about predicate in our example query looks like `about(.//paragraph, -children ''beautiful flowers'))` the score computation for the 'p' element 'not containing' the term 'children' (negative score computation) looks like:

$$\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \not\sqsupset_p \sigma_{n=\mathrm{children},t=\mathrm{word}}(C) \tag{3.7}$$

To achieve the transparency in instantiating retrieval models the score computation is encapsulated in the abstract score computation functions: $f_{\sqsupset}(r_1, R_2)$ and $f_{\not\sqsupset}(r_1, R_2)$, as depicted in the definition of these operators in Table 3.2. Abstract functions $f_{\sqsupset}(r_1, R_2)$ and $f_{\not\sqsupset}(r_1, R_2)$, applied to a region r_1 from the left operand and the right operand region set R_2, should result in the numeric value that specifies the relevance of the region (element) r_1 given the (term) regions in R_2 that r_1 contains. The exact specification of the retrieval model is left to the final instantiation and it could be based on the well-known flat text IR models, as we illustrate in the next chapter.

Score combination operators

For simplicity we do not make a distinction between the low-level and high-level score combination operators in the basic SRA operator set. We only make a distinction between the AND and OR score combination. The former distinction would result in the introduction of variants of the $\sqcap_p$ and $\sqcup_p$ operators. These variants would have exactly the same definition as $\sqcap_p$ and $\sqcup_p$ given in Table 3.2, except that they would allow for different instantiation of abstract operators $\otimes$ and $\oplus$ (e.g., $\otimes'$ and $\oplus'$).

The two abstract operators ($\otimes$ and $\oplus$), used in the definition of score combination operators, specify how scores are combined in the corresponding regions from the left and right operand. The abstract operator $\otimes$ specifies how scores are combined when AND query combination is modeled. The operator that specifies the region AND score combination ($\sqcap_p$) models the combination of scores that have the same region bounds and name and type attributes in the left and right operand. Other regions are not included in the result region set.

The operator $\oplus$ defines score combination in an OR expression, denoted in SRA with $\sqcup_p$. The result region set consists of regions present exclusively in one of the operands (left or right) and regions that are common to both operands, i.e., regions that have the same start, end, name, and type attributes. The score for the latter is determined by the instantiation of score combination operator $\oplus$, while for the former the score is the same as in the existing regions.

Having defined selection, relevance score computation, and score combination operators, we can express the combination of terms in the first *about* expression from our example NEXI query in SRA as follows:

$$\begin{aligned}(R_b \quad :=) \quad & ((\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{children},t=\mathrm{word}}(C)) \\ & \sqcap_p \ (\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{beautiful},t=\mathrm{word}}(C))) \\ & \sqcap_p \ (\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{flowers},t=\mathrm{word}}(C))\end{aligned} \tag{3.8}$$

or if we assume the OR combination among terms in the query as:

$$\begin{aligned}(R_c \quad :=) \quad & ((\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{children},t=\mathrm{word}}(C)) \\ & \sqcup_p \ (\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{beautiful},t=\mathrm{word}}(C))) \\ & \sqcup_p \ (\sigma_{n=\mathrm{p},t=\mathrm{node}}(C) \sqsupset_p \sigma_{n=\mathrm{flowers},t=\mathrm{word}}(C))\end{aligned} \tag{3.9}$$

Similarly, the high-level AND and OR score combination among two different *about*s can be expressed using $\sqcap_p$ and $\sqcup_p$ operators. In our example query we need to combine the scores of 'paragraph' and 'video' elements in the first predicate. However, the combination is only possible after propagating the scores of these elements to the common ancestor 'story' element.

Score propagation operators

The operators $\blacktriangleright$ and $\blacktriangleleft$ specify propagation of scores to the containing or contained elements, respectively. Thus, the abstract function $f_{\blacktriangleright}(r_1, R_2)$ specifies how scores are propagated upwards while abstract function $f_{\blacktriangleleft}(r_1, R_2)$ specify the downwards score propagation. Similarly to operators $\sqsupset_p$ and $\sqsubset_p$, the result region set consists again of regions that have the same starting, end, name, and type attributes as regions in the left operand. The region score is determined by the instantiation of the two abstract functions.

When propagating scores, the functions $f_{\blacktriangleright}(r_1, R_2)$ and $f_{\blacktriangleleft}(r_1, R_2)$ use the values of attributes of regions from the left operand, and the attribute values of contained or containing regions from the right operand. Such propagation of scores from the contained or containing regions should include some form of normalization as the scores of the resulting regions might again be combined with other region scores using score combination operators.

As an illustration of score propagation, in our example query we should propagate the scores from the paragraph regions containing terms 'children', 'beautiful', and 'flowers' (denoted with $R_{b/c}$) to the containing 'story' element, and then down to the contained 'image' element. This can be express as follows:

$$\sigma_{n=\text{image},t=\text{node}}(C) \blacktriangleleft (\sigma_{n=\text{story},t=\text{node}}(C) \blacktriangleright R_{b/c}) \tag{3.10}$$

SRA query expression

Having introduced all the operators following the four elementary structure retrieval requirements, we can now transform the complete example user query (repeated below) into an SRA expression.

```
//story[about(.//paragraph, children beautiful flowers) or
   about(.//video, garden)]//image[about(., flowers garden)]
```

This expression illustrates the elegance of expressing structured search in SRA. To shorten the SRA expression we use R^n_{name} for the selection on element nodes with the region name attribute *name*, i.e., $R^n_{name} := \sigma_{n=name,t=\text{node}}(C)$, and R^w_{name} for the selection on word *name*, i.e., $R^w_{name} := \sigma_{n=name,t=\text{word}}(C)$. The SRA expression is shown below.

$$\begin{aligned}(R_q \quad :=) \quad & ((R^n_{image} \blacktriangleleft ((R^n_{story} \blacktriangleright (((R^n_p \sqsupset_p R^w_{children}) \\ & \sqcap_p (R^n_p \sqsupset_p R^w_{beautiful})) \sqcap_p (R^n_p \sqsupset_p R^w_{flowers}))) \\ & \sqcup_p (R^n_{story} \blacktriangleright (R^n_{video} \sqsupset_p R^w_{garden})))) \sqsupset_p R^w_{flowers}) \\ & \sqcap_p \\ & ((R^n_{image} \blacktriangleleft ((R^n_{story} \blacktriangleright (((R^n_p \sqsupset_p R^w_{children}) \\ & \sqcap_p (R^n_p \sqsupset_p R^w_{beautiful})) \sqcap_p (R^n_p \sqsupset_p R^w_{flowers}))) \\ & \sqcup_p (R^n_{story} \blacktriangleright (R^n_{video} \sqsupset_p R^w_{garden})))) \sqsupset_p R^w_{garden})\end{aligned} \tag{3.11}$$

By comparing this query expression to the relational algebra expressions depicted in Equation 2.27 (given in Section 2.4.1) we can see a great resemblance. However, there are some essential differences. In the relational expression we use a combination of relational operators to express containment relation, and most importantly this combination only specifies the strict containment. In SRA this expression encapsulates also the relevance ranking mechanism. This shows that the SRA is more suitable for expressing structured IR search. Furthermore, the SRA operators are defined in such a way that they are not bound to a specific retrieval model instantiation, allowing transparent instantiation of retrieval models following the four elementary retrieval requirements, and providing the desired *retrieval model independence.*

3.2.5 Variants of score region algebra

In our approach to structured retrieval we extended region algebra approaches to be able to express relevance scoring mechanisms. We introduce new region attributes to model structured data and new operators that are extensions of containment operators from other region algebra approaches. These operators model how scores are assigned to different regions based on their containment relations and region attribute values. The question is why we did choose this set of operators when others are possible. There are several reasons that are explained in the sequel, but before we discuss them we present two alternative approaches for specifying score region algebra.

Model-specific SRA

One approach is to define SRA operators in such a way that the abstract functions and abstract operators are fixed. To enable instantiations of different retrieval models (for different domains) different SRA query plans should be generated. In [97] this is explained for the statistical language modeling approach, and demonstrated on three different retrieval approaches: for flat text search, for video shot retrieval using speech transcripts, and for web retrieval using page priors.

In this case the abstract function used for score computation is implemented as given in Equation 3.12.

$$f_{\sqsupset}(r_1, R_2) = \frac{|\{r_2 | r_2 \prec r_1\}|}{r_1.e - r_1.s - 1} \tag{3.12}$$

Abstract operators $\otimes$ and $\oplus$ are implemented as product and sum. Also, additional operator $\sqsubset_p$ is defined similar to $\sqsupset_p$, except that it returns regions from the left operand contained in regions from the right operand or regions from the left operand that have the same region bounds as regions in the right operand. The score attribute is defined through the $f_{\sqsubset}(r_1, R_2)$ function defined in Equation 3.13.

$$f_{\sqsubset}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2, r_1 \prec r_2} r_2.p \tag{3.13}$$

To enable the usage of retrieval model parameters the scaling mechanism is introduced, implemented as $\circledast$ operator defined as follows:

$$R \circledast num = \{(r.s, r.e, r.n, r.t, r.p \cdot num) | r \in R\} \tag{3.14}$$

For illustration, the first *about* clause from the simplified example query, i.e., `about(.//paragraph, children beautiful flowers)`, can be expressed in this variant of SRA as follows (we used the shorthand notation for the element and term selection):

$$
\begin{array}{rcl}
\sigma^n_{\mathrm{p}}(C) & \sqsubset_p & \\
((((\sigma^n_{\mathrm{p}}(C) \sqsupset_p \sigma^w_{\mathrm{children}}(C)) \circledast \lambda) & \sqcup_p & ((\sigma^n_{\mathrm{Root}}(C) \sqsupset_p \sigma^w_{\mathrm{children}}(C)) \circledast (1-\lambda))) \\
 & \sqcap_p & \qquad (3.15) \\
(((\sigma^n_{\mathrm{p}}(C) \sqsupset_p \sigma^w_{\mathrm{beautiful}}(C)) \circledast \lambda) & \sqcup_p & ((\sigma^n_{\mathrm{Root}}(C) \sqsupset_p \sigma^w_{\mathrm{beautiful}}(C)) \circledast (1-\lambda)))) \\
 & \sqcap_p & \\
(((\sigma^n_{\mathrm{p}}(C) \sqsupset_p \sigma^w_{\mathrm{flowers}}(C)) \circledast \lambda) & \sqcup_p & ((\sigma^n_{\mathrm{Root}}(C) \sqsupset_p \sigma^w_{\mathrm{flowers}}(C)) \circledast (1-\lambda)))
\end{array}
$$

As can be seen, even for such a simple query, the SRA expression is quite complex. Additionally, the instantiation of different retrieval models that involve more statistical information and more parameters, would result in even more complex queries. This is in contrast to our desire to keep the framework simple and also to enable transparent specification of retrieval models when defining region algebra operators (as pointed out later in discussion).

Coarse SRA

Looking from the implementation point of view, it is more appropriate to treat the search on terms within one search or answer element as one operation than to treat the search in isolation per query term. In such a scenario, the expensive containment operation would have to be performed only once on a set of element regions and on the union of sets of all the search term regions. This would simplify the implementation and give more opportunities for faster query execution.

Furthermore, in this way we would avoid the introduction of selection operator on terms. This would only allow the specification of operators that produce the retrievable units, i.e., structured elements. Looking at the nine operators given in Table 3.2, the operator set would be reduced. The score propagation and score combination operator definitions would remain the same, except that the score combination operators would only define the high-level score combination. The low-level score combination operator would be hidden in the new scoring operator explained below.

Instead of relevance score computation operators we would have a complex high-level element selection operators that fuse the function of term and element selection operator ($\sigma_{n=name,t=type}(R)$) and score computation operators ($\sqsupset_p$ and $\not\sqsupset_p$). These operators would involve many more parameters and include more complex manipulation on region attribute values. If we denote these operators with α we can define positive and negative element score computation as depicted in Equations 3.16 and 3.17.

$$
\begin{array}{rcl}
\alpha^{\sqsupset\{tm_1,tm_2,...,tm_n\}}_{n=name,t=type}(R) & = & \{(r.s, r.e, r.n, r.t, f_{\alpha,\sqsupset}(r, tm_1, tm_2, ..., tm_n)) \\
 & & \mid r \in R \ \wedge \ r.n = name \ \wedge \ r.t = type\} \qquad (3.16)
\end{array}
$$

$$\begin{aligned}\alpha_{n=name,t=type}^{\not\sqsupset\{tm_1,tm_2,...,tm_n\}}(R) \quad &= \quad \{(r.s, r.e, r.n, r.t, f_{\alpha,\not\sqsupset}(r, tm_1, tm_2, ..., tm_n)) \\ &\qquad \mid r \in R \ \wedge \ r.n = name \ \wedge \ r.t = type\} \end{aligned} \tag{3.17}$$

Such specification of relevance score computation seems to be in accordance with the NEXI query specification. For example, the first *about* clause in the simplified example query can be expressed in such a variant of SRA as:

$$\alpha_{n=\text{p},t=\text{node}}^{\sqsupset\{\text{children, beautiful, flowers}\}}(C)$$

However, important reasons exist that are not in favor of choosing such operator granularity, but the one represented in Table 3.2. They are explained below.

Discussion

The operators in our SRA, given in Table 3.2, are specified in such a way that they:

- follow the identified elementary structured retrieval requirements
- support retrieval model independence
- support content description independence
- provide good balance between operator complexity and the number of operators needed to model structured retrieval
- provide an opportunity for using operator properties for query rewriting and optimization.

First of all, in the original SRA, the operators are defined in such way that they are in accordance with the identified four elementary retrieval requirements. We define operators that model the selection of different entities, element relevance score computation, score combination, and score propagation. However, in the model-specific SRA the score computation aspect is expressed as a retrieval model specific combination of several entity selection and score computation operators, while in the coarse SRA we loose the explicit modeling of entity selection structured retrieval requirements.

Dropping the entity selection requirement in the coarse SRA might seem as a small problem when discussing only text retrieval where operator α is defined on structured elements and a set of terms. However, such operator definition becomes problematic if we consider that it should also model the search on phrases, term modifiers, synonyms, etc., in addition to search on single words. Furthermore, the complexity of the operator α grows if it also defines, e.g., image and video search besides textual search. Having such a coarse specification of retrieval process would make the full analysis of the structured retrieval subtasks more difficult.

On the other hand, for model-specific SRA different model implementations would require detailed analysis of the retrieval model specification as well as translation of query expressions to different SRA query plans for each retrieval model. The model-specific SRA does not support the retrieval model independence and content description independence. Although the operator set is fixed in model-specific SRA for text search, we would be forced to generate different query plans for each model we want to implement. Furthermore, when incorporating different document representation (features), such as video and audio representations, new operators need to be introduced and balanced with the existing ones. Again, different query plans would have to be generated.

In the coarse SRA, the retrieval model, except high level score combination and score propagation, is modeled as one black box. Everything related to the score manipulation is hidden in the $f_{\alpha,\sqsupset}$ and $f_{\alpha,\not\sqsupset}$ functions: treatment of elements and terms, statistics used for the specification of the functions, etc. This is not in accordance with our desire to have the retrieval process easily monitored and analyzed (see the analysis in Chapters 5 and 6). Although content description independence can be supported in coarse SRA it would result in drastically increased complexity of the black box functions representing different retrieval models.

By isolating the entity selection aspect, as we do in the original SRA operator set, different entities, such as words, structured elements, video, and audio content can be selected and treated in the same SRA framework (see Chapter 7). The selection is performed disregarding the way how different types of document content are modeled, enabling in such a way content description independence.

Therefore, while in model-specific SRA we would end-up with the complex model-specific query plans, in coarse SRA we would end-up with too complex scoring functions $f_{\alpha,\sqsupset}$ and $f_{\alpha,\not\sqsupset}$. For example, as the operators in coarse SRA encapsulate more terms and optionally other parameters, such as term modifiers and synonyms, a huge number of functions need to be defined to cope with all these variants. Even though for the same retrieval model that specifies the relevance of an element with respect to a term, where different score combination implementation is used, different functions $f_{\alpha,\sqsupset}$ and $f_{\alpha,\not\sqsupset}$ need to be specified. Such an explosion of functions would be against our desire to keep the framework simple and extensible, pertaining the retrieval model independence.

Therefore, having such classification of operators in the original SRA, depicted in Table 3.2, a good balance between the number of operators that model structured retrieval and operator complexity is achieved. The operator set consists of only nine operators that are used to model most of the user requests over structured documents. Each operator expresses one concept within structured retrieval requirements.

Finally, such specification of operators in the original SRA (given in Table 3.2) provides the opportunity for using operator properties for query rewriting and optimization. As operator properties depend on the instantiation of retrieval models we discuss them in the next chapter.

At the end of this section we present the relation among the granularity of the operators in variants of SRA. In the coarse SRA, high-level element selection operator can be considered as a generalization of the original SRA selection and score computation operators. This is similar to the relational algebra, where the frequent combination of selection operator and Cartesian product is replaced with the join operator. This might be interesting for having efficient implementations of retrieval models.

Similarly, model-specific SRA operators are a specialization of the original SRA operators, where retrieval models are instantiated without using the abstract operators and abstract functions in the algebra. The decision on retrieval model used, i.e., the proper SRA query plan, is done at higher level. This might be useful for providing insight in the specific retrieval model functionality on a particular retrieval task, but not for analyzing structured retrieval.

The coarse and model-specific SRA are two possible directions that we can take in our future research. However, this should be done when we understand better the process of structured retrieval.

3.3 SRA: opportunities and limitations

In this section we present good and bad features of score region algebra. The features are described along three scenarios: (1) extensions of region data model to support more information describing structured documents, (2) specification of new retrieval models in SRA, and (3) support for data extracted from non-textual descriptions of document components.

3.3.1 Region model extensions

Looking at the definition of basic region algebra operators given in Table 3.2 we can see that they are composed of two parts. The first part either tests the name and type attributes of regions ($\sigma_{n=name,t=type}(R)$), test their starting and end attributes on containment ($R_1 \sqsupset R_2$ and $R_1 \sqsubset R_2$) or equality ($R_1 \sqcap_p R_2$ and $R_1 \sqcup_p R_2$), or copy the region beginning, end, name, and type attributes to the result region set ($R_1 \sqsupset_p R_2$, $R_1 \not\sqsupset_p R_2$, $R_1 \blacktriangleright R_2$, and $R_1 \blacktriangleleft R_2$). The second part is related to score manipulation and can involve all attributes of the regions in the left and in the right operand. As the scoring mechanism employing these attributes is hidden in the abstract functions and operators, the region attribute set can easily be extended by introducing new attributes, without affecting the operator definition.

For example, if we would like to model parent-child relationship in the SRA data model, we might introduce a parent attribute, denoted with o, that would store the starting position of the parent node in the SRA data model. Therefore, the region data model needs to be extended with a sixth attribute and would look like: $r = (r.s, r.e, r.o, r.n, r.t, r.p)$. In such case the SRA region set for the example XML document given in Figure 3.1 would look like ($-$ denotes that a region has no parent region):

STORY	−−	$\{(0, 1067, -, \text{story}, \text{node}, 1.0), ...$
ID	−−	$(1, 2, -, \text{id}, \text{attr\_name}, 1.0), ...$
78	−−	$(2, 2, -, 78, \text{attr\_val}, 1.0), ...$
TITLE	−−	$(3, 7, 0, \text{title}, \text{node}, 1.0), (15, 18, 12, \text{title}, \text{node}, 1.0), ...$
THE	−−	$(4, 4, 3, \text{the}, \text{word}, 1.0), ..., (1046, 1046, 1043, \text{the}, \text{word}, 1.0), ...$
GARDEN	−−	$(17, 17, 12, \text{garden}, \text{word}, 1.0), (52, 52, 20, \text{garden}, \text{word}, 1.0), ...$
P	−−	$(46, 188, 0, \text{p}, \text{node}, 1.0), ..., (1043, 1066, 0, \text{p}, \text{node}, 1.0),$
...	−−	$... \}$

Although this new information item would not override the specification of the operators, it would make possible the introduction of new operators, as well as the alternation of the specification of abstract functions and abstract operators that define the retrieval model. For example, operators can be defined for selecting child or parent regions from arbitrary region sets. The definition of these operators is given in Equations 3.18 and 3.19.

$$\omega_{child}(R_1) = \{r \mid r \in C \wedge r_1 \in R \wedge r.o = r_1.s\} \tag{3.18}$$

$$\omega_{parent}(R_1) = \{r \mid r \in C \wedge r_1 \in R \wedge r_1.o = r.s\} \tag{3.19}$$

Similarly, other information items can be added to the SRA attribute set, such as nesting level of XML elements, relative position of terms and elements in a containing element, path leading to an element/term. Element and term paths can be useful for computing additional element and term statistics, and therefore specifying new retrieval models. On the other hand, element and term positions can be used for introducing proximal operators, and element and term level can be used for introducing new parameters in the retrieval model specifications (presented in Section 7.1).

Although SRA data model can easily be extended to express arbitrary information item extracted from structured documents, these extensions introduce additional complexity in specifying retrieval models. However, having cleanly defined basic SRA data model and operators, as well as abstract functions and abstract operators, this extensions can be considered as a special purpose SRA modification (similarly as SQL implements more than basic relational operators).

3.3.2 Introduction of new retrieval models

SRA framework is designed for supporting the specification of various retrieval models without affecting the SRA data model and operator definitions. The only assumption is that the retrieval model should be specified following the four elementary structured retrieval requirements. This is not a hard request as, e.g., in most flat retrieval models score computation and score combination aspects can be isolated. Score computation looks at term-element or term-document pairs in isolation and computes the relevance score of an element or a document. Score combination on the other hand specifies how these isolated term-element pair scores can be combined.

The retrieval model instantiation that follows the four elementary retrieval requirements is the main topic of the next chapter and we do not discuss it further here. We just illustrate what are three major drawbacks of SRA when specifying various retrieval models. These are:

- it is difficult to express retrieval models that utilize term proximity for element relevance score computation
- it is difficult to express operations that include path manipulation
- the choice of the default region score can have negative influence on retrieval model effectiveness.

Since score computation aspect in SRA is modeled per term basis, after computing the relevance score of an element the information about the position of the contained term is lost. It cannot be used for modifying scores of an element that, e.g., contains two terms close to each other. Therefore, new complex operators have to be defined that can handle term proximity expressions, such as the one specified in Equation 3.16 for the coarse SRA ($\alpha^{\sqsupset\{tm_1,tm_2,\ldots,tm_n\}}_{n=name,t=type}(R_1)$), or the ones that define phrase search (see [144]).

In SRA, it is in many cases not possible to say whether two elements are on the same absolute path from the root node or not. For example, the NEXI expression `//story//title` would give 'title' elements from which some of the elements are on one path (e.g., `/story/title`) and some are on different paths (e.g., `/story/title` and `/story/image/title`). Even with the introduction of a region attribute that specifies the path leading to an element or a term, incorporating paths into retrieval models would involve string manipulation on strings representing absolute paths to structured elements. In other words, the scoring function would have to match substrings to determine whether two paths are the same or how similar they are.

The choice for the default region score is left open in SRA. The administrator/user can specify different default scores for the same or different retrieval models. While this leaves great flexibility for the retrieval model implementation, many retrieval models will make sense only with a particular choice for the default region score. For example, if a retrieval model is based on multiplying region scores, having the default region score 0 would not be appropriate. Similarly if retrieval model sums the default scores, the default score of 1 might be a bad choice. Therefore, the choice of default region score can have significant (negative) effect on the effectiveness of retrieval models instantiated in score region algebra.

3.3.3 Inclusion of other media

The basic score region algebra data model and operator set are defined for search in textual documents. However, documents can contain other non-textual content, such as video, audio, or images, as discussed in Section 2.1.3. Although not given in the basic operator set score region algebra can easily be extended to support selection of different types of document components and their incorporation in the model.

Multimedia document components can be selected based on the content descriptions in two ways. The first is by simply matching the query terms with the textual description of multimedia content, in case it is available. The second one is matching multimedia features (using some multimedia IR model) with the features of multimedia component given as a sample (see NEXI example and discussion in Section 1.2.3). The modeling of video clips and images and their incorporation in SRA framework is explained in Chapter 7.

The potential problem for such inclusion of different domain descriptions might be that each type of description would result in a new operator (or a modification of the existing one) for supporting it. Furthermore, this operator needs to encapsulate the multimedia retrieval approach and produce the result which could be further exploited in score region algebra. Although this is not trivial, in Section 7.3 we show that it is possible for the image retrieval models.

3.4 Summary

This chapter identifies four elementary requirements for the development of our structured retrieval system: entity selection, relevance score computation, score combination, and score propagation. These requirements are illustrated on an example NEXI query. After identifying the requirements, we specify the framework used for structured IR that follows these requirements. The specification is done by extending region algebra approaches to handle relevance scoring mechanism. The resulting new algebra is called score region algebra (SRA). SRA is based on a simple data model, defined on a set of regions, where each region has its starting position, end position, name, type, and score attributes. The elementary set of operators consists of nine operators that model the basic operations needed for structured retrieval following the four elementary retrieval requirements.

SRA operators are defined abstracting away from the exact retrieval model implementation and enabling the instantiation of different retrieval models without affecting the score region algebra framework. The choice for such specification of the algebra's data model and operators is justified by comparing it to other possible variants of SRA. Furthermore, the potential of using the SRA framework for modeling structured retrieval is illustrated and some of its deficiencies are discussed at the end of the chapter.

Chapter 4

Transparent Retrieval in Structured Documents

This chapter discusses the features of our three-level database system, called TIJAH, and explains its main components. Then, the instantiation of different retrieval models in score region algebra, following the four elementary retrieval requirements, is explained. The chapter ends with a short overview of SRA operator properties using different retrieval model instantiations.

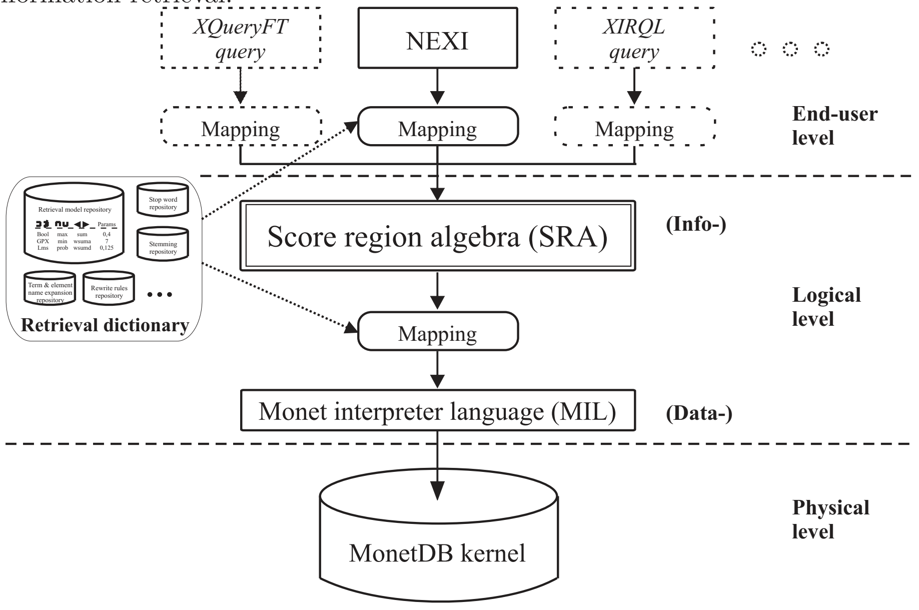
This chapter is partially based on papers published (1) in the Proceedings of the Joint Workshop on XML, IR and DB in conjunction with the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval [140], (2) in the Information Retrieval Journal [123], and (3) in the Proceedings of the 14th ACM International Conference on Information Knowledge and Management (CIKM) [138].

4.1 TIJAH retrieval system architecture

In the development of the structured retrieval system we followed the database approach. Thus, the TIJAH [17] structured IR system follows a traditional three-level database architecture, consisting of end-user, logical, and physical level, where respectively external, conceptual, and internal schema are instantiated (see Figure 2.2 in Chapter 2). The architecture of the TIJAH system is depicted in Figure 4.1, following the designers view on databases by Senko [200].

The central part of the system is the transparent logical level based on score region algebra (SRA), specified in the previous chapter. The transparency is reflected by the ability to instantiate different retrieval models without affecting the specification of the logical operators, while keeping the same query language (or query languages) at the end-user level and with the physical implementation using an arbitrary dedicated IR engine or a DBMS. This transparency is supported by a retrieval dictionary (depicted with a rectangle area on the left part of Figure 4.1) that guides the mapping processes for the end-user to SRA transformation and SRA to physical plan transformation. The main component of the retrieval dictionary is a retrieval model dictionary that stores details of different retrieval models that are implemented at the physical level.

Figure 4.1: TIJAH: Architecture of a three-level database system for structured information retrieval.



For the end-user level, i.e., external schema, TIJAH uses Narrowed Extended XPath I (NEXI) [209] view on structured data and the NEXI query language. The choice was driven by the simplicity of the NEXI syntax, its power in expressing the most important structured IR tasks, and as NEXI query language is the query language used in INEX [74]. As TIJAH follows the database approach, i.e., supports logical data independence, other query languages, such as XIRQL [72] or XQuery full-text [7], could equally likely be implemented on top of the score region algebra (as shown with dashed boxes in Figure 4.1).

For the physical implementation we use the MonetDB [19] database kernel, as it is an efficient database system that can also be used for fast prototyping. The communication between SRA and the MonetDB kernel is established through the Monet interpreter language (MIL) at the datalogical level. The details of the system's components, along end-user, logical, and physical levels, are explained below.

4.1.1 End-user level

At the end-user level a user query is transformed (mapped) into the logical algebra query plan. In TIJAH, a NEXI query is first processed and then transformed into a logical query plan (see Figure 4.2). A fully automatic approach is developed for processing NEXI queries and for the generation of logical query plan forwarded to the logical level.

As explained in Section 2.4.2, NEXI allows only descendant steps from XPath 1.0 [37] and introduces an *about* clause that specifies which part of the document should be searched and what are the query terms. Furthermore, NEXI distinguishes two types of queries:

- Content-only (CO) queries that correspond to flat text list-of-term queries. However, while list-of-term queries express the need like “find documents that are about the query terms”, CO query expresses the need like “find the appropriate document components that are about the query terms”.

- Content-and-structure (CAS) queries denote queries in which a user specifies where in a document he would like to search for information and what would he like to get as an answer.

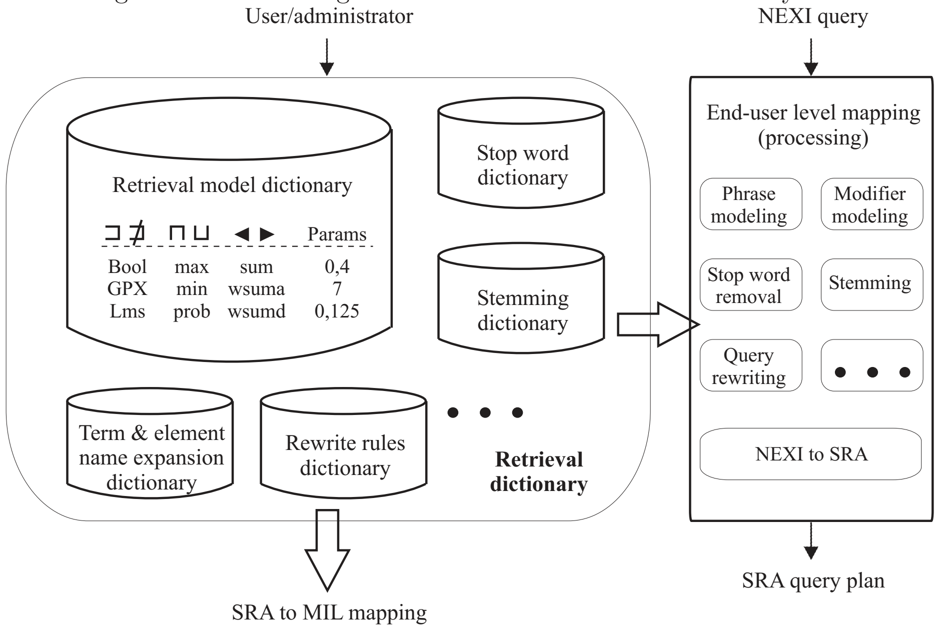
In this thesis we focus on structured retrieval where the user explicitly states the structured constraints in the query. Therefore, the emphasis is on CAS queries. Also, in the TIJAH system CO queries are transformed into CAS queries using rewriting rules within the query rewriting processing unit (see below).

End-user level processing

The structure of the end-user query translator is depicted in Figure 4.2. The end-user level of the TIJAH system consists of a number of processing units among which the most important ones are: phrase modeling unit, modifier modeling unit, stop word removal unit, stemming unit, query rewriting unit, and NEXI to SRA mapping unit. For the processing of NEXI queries, the data from the retrieval dictionary is consulted. The most important components of retrieval dictionary are the retrieval model dictionary, the stop word and stemming dictionary, the term expansion dictionary, the element name expansion dictionary, and the rewrite rules dictionary. Each of these dictionaries stores the necessary data or rules for the functioning of query processing units, as well as for the generation of MIL query plans out of SRA query plans, as discussed in the next section. Other processing units can easily be plugged in at the TIJAH end-user level. The same is true for other dictionaries in the retrieval dictionary.

End-user level processing units are responsible for supporting the traditional IR system query processing (see e.g., Introduction in [12]) and for extending the query according to the user/administrator specification and based on the content of the rewrite rules dictionary. Before explaining each of the processing units we first give a short overview of the components of the retrieval dictionary, with the emphasis on the central component – the retrieval model dictionary.

Figure 4.2: Processing at the end-user level of the TIJAH system.



Retrieval dictionary

The data from the *retrieval model dictionary* form the input to the end-user processing and for the infological (SRA) to datalogical (MIL) mapping. The retrieval model dictionary in the TIJAH system stores the options for the instantiation of different retrieval models following the four elementary structured IR requirements. Thus, it stores what kind of entity selection is going to be performed, what are the options for score computation, combination, and propagation functions. It also stores various parameters necessary for the implementation of retrieval models.

An example retrieval model configuration file can be seen in Figure 4.3. Here the first row indicates the selected options for the retrieval model setting. The names in the row below it explain what each number/text in the first row represents, e.g., score computation (`MODEL`), score combination (`AND` and `OR`), model parameters (`P1`, `P2`, and `P3`). The rows below the dashed line depict the possible values for each option in the retrieval model dictionary. For example, OR score combination can be implemented as sum (parameter value 1), maximum (parameter value 2), probabilistic sum (parameter value 3), etc.

In the example configuration file from the retrieval model dictionary, given in Figure 4.3, language model (`LMS`) is used for the score computation function, OR

Figure 4.3: An example of a text file describing the possible parameters of a retrieval model in the retrieval model dictionary.

```
1    3        1      1      1      1      FALSE   04fm001   1     1     0.5   0.75  5     1     shot    0.8

NUM  MODEL    OR     AND    UP     DOWN   E_CLASS EXP_CLASS STEM  SIZE  P1    P2    P3    LTYPE CONTEXT EXTRA
-------------------------------------------------------------------------------------------------------------
=1   BOOL 1   SUM 1  PROD 1 SUM 1  SUM 1  FALSE   string    NO 1  ENT 1 float float int   NO 1  string  float
     LM 2     MAX 2  MIN 2  AVG 2  AVG 2  TRUE              YES 2 TRM 2                   LP 2
     LMS 3    PROB 3 SUM 3  WSD 3  WSD 3                                                  LN 3
     TFIDF 4  EXP 4  EXP 4  WSA 4  WSA 4
     OKAPI 5  PROD 5 MAX 5
     GPX 6    MIN 6  PROB 6
     LMA 7
     LMSE 8
     LMVFLT 9
     LMVLIN 10
```

and AND score combination functions are implemented as sum and product (`SUM` and `PROD`), the value of the parameter λ is 0.5 (`P1` which is of float type), etc.

The *stop word dictionary* contains a stop word list, i.e., a list of words that most frequently occur in one language. Since we experimented with English and Dutch collections, the stop word dictionary contains English (429 words) and Dutch (131 words) stop word lists.

The *stemming dictionary* contains rules needed for finding the stem of a word. We use the standard Porter stemmer [172] for English and a Dutch variant developed by Kraaij and Pohlmann [113].

The *term and element name expansion dictionary* supports query term expansion as well as retrieval from heterogeneous collections and vague search where element names need not to be matched strictly. The element name expansion dictionary contains a number of element name synonyms that can be used for vague element name search. For example, the search on 'sections' can be transformed into the search on 'sec', 's', 'ss1', etc. In our experiments we use INEX equivalent element names [209] and these are stored in the term and element name expansion dictionary. We do not use any term expansion dictionary for our experimentation. However, it can easily be incorporated using, e.g., WordNet [147].

The *rewrite rules dictionary* stores the rewriting rules that are used to rewrite the NEXI queries at the end-user level. It stores the rules on how terms can be placed in different search contexts (search elements), or how structured constraints can be loosen (see [145] for more details).

End-user processing units

The *phrase modeling unit* handles phrases in the NEXI query. The user (or administrator) can specify whether phrases should be considered as phrases or as a set of terms in the query, based on the content of the retrieval model dictionary. Therefore, the phrase modeling unit either removes the '`"`' signs (no phrases) or leaves them in.

Whether term and phrase modifiers ('+', and '−') should be considered during query execution and how they should be modeled is regulated in the *modifier modeling unit*. Based on the content of the retrieval model dictionary they can be modeled as strict or vague modifiers. Although phrases and modifiers are supported in TIJAH, we report only the results without using phrases and term modifiers in the thesis. More details on modeling phrases and modifiers in TIJAH can be found in [144].

The standard IR query processing, consisting of query stop word removal and stemming, is done by the *stop word removal unit* and *stemming unit*. Our stop word and stemming dictionary supports English and Dutch stop word removal and stemmers, but other stemmers, or stemmers for other languages can easily be added to the system through the stemming dictionaries. This is also true for other stop word lists.

The end-user *query rewriting unit* distinguishes between NEXI content-only and content-and-structure query expansions. Since the score region algebra is designed to work on structured queries, NEXI CO queries are transformed into CAS queries according to the user specification. For instance, an example CO query that expresses the search for the appropriate element about children playing in a garden with beautiful flowers: `children beautiful flowers garden` is rewritten into the following NEXI CAS query:

```
//*[about(., children beautiful flowers garden)]
```

Here '`*`' denotes the selection of arbitrary XML nodes.

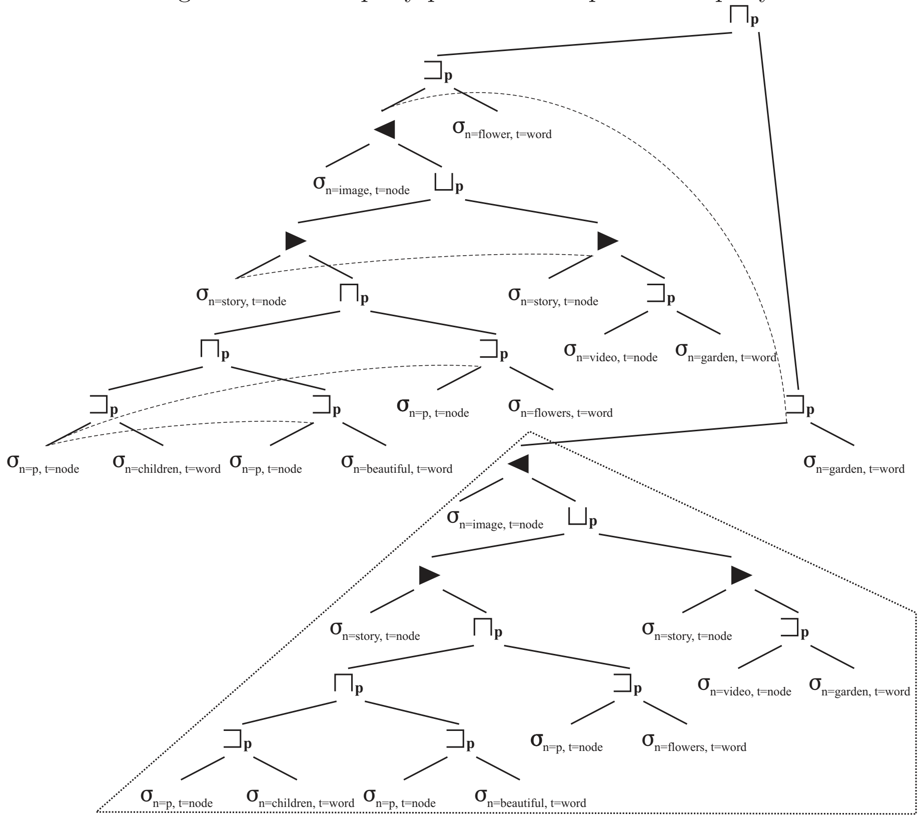
On the other hand, based on the content of rewrite rules dictionary, CAS query rewriting specifies the relaxing of structured constraints in the *about* clause and interchanging the terms among different *about* clauses. Some simple rules are defined to enable elementary CAS query rewriting but they are beyond the score of this thesis. More details can be found in [139].

Finally, the *NEXI to SRA unit* is a parser-like processing unit that transforms the processed NEXI query into the score region algebra query plan. In this transformation each entity in the query is translated into one selection operator, and these operators then form an 'input' to the score computation operators for each term in the *about* clause. Then they are used as an 'input' to implicit and explicit score combination operators, and also to score propagation operators that specify the modeling of NEXI descendant steps (i.e., `//`) in the score region algebra.

4.1.2 SRA at (info)logical level

At the logical level, structured information retrieval is implemented using the score region algebra discussed in the previous chapter. The main characteristic of the SRA operator set is the transparent instantiation of scoring functions. The possible instantiations of abstract functions and operators are presented in the next section. However, due to transparency, without knowing the exact implementation of abstract functions and abstract operators we can generate the SRA query plan.

Figure 4.4: SRA query plan for example NEXI query.


$\sqcap_p$
$\sqsupset_p$
$\sigma_{n=flower, t=word}$
$\sigma_{n=image, t=node}$
$\sqcup_p$
$\sigma_{n=story, t=node}$
$\sqcap_p$
$\sigma_{n=story, t=node}$
$\sqsupset_p$
$\sqcap_p$
$\sqsupset_p$
$\sigma_{n=video, t=node}$
$\sigma_{n=garden, t=word}$
$\sqsupset_p$
$\sqsupset_p$
$\sigma_{n=p, t=node}$
$\sigma_{n=flowers, t=word}$
$\sqsupset_p$
$\sigma_{n=p, t=node}$
$\sigma_{n=children, t=word}$
$\sigma_{n=p, t=node}$
$\sigma_{n=beautiful, t=word}$
$\sigma_{n=garden, t=word}$
$\sigma_{n=image, t=node}$
$\sqcup_p$
$\sigma_{n=story, t=node}$
$\sqcap_p$
$\sigma_{n=story, t=node}$
$\sqsupset_p$
$\sqcap_p$
$\sqsupset_p$
$\sigma_{n=video, t=node}$
$\sigma_{n=garden, t=word}$
$\sqsupset_p$
$\sqsupset_p$
$\sigma_{n=p, t=node}$
$\sigma_{n=flowers, t=word}$
$\sigma_{n=p, t=node}$
$\sigma_{n=children, t=word}$
$\sigma_{n=p, t=node}$
$\sigma_{n=beautiful, t=word}$

For example, the structured NEXI query introduced in the previous chapter:

```
//story[about(.//paragraph, children beautiful flowers) or
      about(.//video, garden)]//image[about(., flowers garden)]
```

results in the query expression given in Equation 4.1 (p stands for the 'paragraph' nodes). In the expression we use the shorthand notation introduced in the previous chapter, i.e., R^n denotes the selection of element node regions while R^w denotes the selection of word regions. The graphical representation (query plan) of the expression is given in Figure 4.4.

$$\begin{aligned}(R_q \quad :=) \quad & ((R^n_{image} \blacktriangleleft ((R^n_{story} \blacktriangleright (((R^n_p \sqsupset_p R^w_{children}) \\ & \sqcap_p (R^n_p \sqsupset_p R^w_{beautiful})) \sqcap_p (R^n_p \sqsupset_p R^w_{flowers}))) \\ & \sqcup_p (R^n_{story} \blacktriangleright (R^n_{video} \sqsupset_p R^w_{garden})))) \sqsupset_p R^w_{flowers}) \\ & \sqcap_p \\ & ((R^n_{image} \blacktriangleleft ((R^n_{story} \blacktriangleright (((R^n_p \sqsupset_p R^w_{children}) \\ & \sqcap_p (R^n_p \sqsupset_p R^w_{beautiful})) \sqcap_p (R^n_p \sqsupset_p R^w_{flowers}))) \\ & \sqcup_p (R^n_{story} \blacktriangleright (R^n_{video} \sqsupset_p R^w_{garden})))) \sqsupset_p R^w_{garden})\end{aligned} \tag{4.1}$$

A query plan from Figure 4.4 would be very inefficient as it contains a number of duplicate subtrees in a query tree. For example, the whole subtree on the left part of the $\sqsupset_p$ operator in the bottom part of Figure 4.4 (denoted with dotted rectangle) is a copy of the upper left part of the figure. Thus, an elementary optimization step is performed at the logical level, transforming the SRA query tree into a graph (presented with dashed lines in Figure 4.4). The other optimization types, such as using SRA operator properties discussed later in this section, can also be implemented at the logical level. As this requires more research and more experimentation on structured retrieval tasks using our score region algebra framework, we left it for future research.

4.1.3 TIJAH internals – MonetDB and MIL

Having the specification of retrieval models, and having the user selecting the retrieval model options from the retrieval model dictionary (see Figure 4.3), the query plan is transformed into calls to Monet interpreter language (MIL) procedures at the datalogical level. For each SRA operator, and for each implementation of abstract functions and abstract operators, different MIL procedures are called. The physical implementation in the TIJAH system is done using MonetDB kernel [19]. Here we explain the basics of the MonetDB system and its interpreter language, illustrating the easiness of transforming SRA query plans into calls to MIL (MonetDB) functions.

Table 4.1: Relational database structures used for storing example XML document (given in Figure 2.3).

void	start	end	name	type
0@0	0	1067	`story`	node
1@0	3	7	`title`	node
2@0	4	6	`nil`	text
...	...	...	...	...
8@0	12	19	`image`	node
9@0	15	18	`title`	node
10@0	16	17	`nil`	text
...	...	...	...	...

void	pos	word
0@0	4	`The`
1@0	5	`Selfish`
2@0	6	`Giant`
...	...	...
23@0	51	`lovely`
24@0	52	`garden`
...	...	...

MonetDB

MonetDB is an open source low-level database system developed at the Institute for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands [19]. It was designed to provide high performance on complex queries against large databases. The primary data structure of MonetDB is a *binary association table – BAT*. A BAT is a main-memory structure that represents the binary relationship between two atomic types. Apart from the usual atomic types, such as bit, char, integer, float, double, string, MonetDB supports *oid* and *void* atomic types. The *oid* atomic type is a unique long integer used as an object identifier. Virtual-oids (*void*s) represent densely ascending column of *oid*s that does not take any storage space. Such virtual-oids are quite handy for fast look-ups.

BATs are the basic storage units on which MonetDB operators are defined. A BAT may hold an unlimited number of binary associations, where the two attributes of a BAT are called head (left) and tail (right). A number of operators are defined and kept in MonetDB modules. These operators define (1) real, integer, and string operators, (2) relational-like operators on BATs, such as join, projection, selection, as well as (3) special purpose operators, such as a tree-merge join for implementing containment relations [122].

Having such two-column (BAT) storage structures, the SRA data model can be translated and the structured collection data can be stored into MonetDB BATs. Using MonetDB operators SRA operators can be implemented. The MonetDB storage structures are illustrated in Table 4.1. In our implementation we use the horizontal fragmentation, explained in Section 2.4.1 and depicted in Table 2.3, for storing structured (XML) data. Therefore, we have three different fragments of a data collection region set (table): the first represents element nodes, the second content words, and the third element attributes.

For illustration, in Table 4.1 we only give BATs formed from element node and term fragments. Although represented as one relational table, the region (relational) table that contains more than two columns is also fragmented into a number of tables (BATs), each containing one *void* head column and one data tail column[1]. The *void* column is used as a region identifier. The BAT that stores (default) region scores does not exist, as in that way we provide the possibility to dynamically set the default score value.

Using the MonetDB set of operators, we can define arbitrary manipulation over BATs. Therefore, the MIL expressions need to specify the manipulation over partitioning of region tables in MonetDB presented in Table 4.1, using relational, real, integer, string, and special purpose MonetDB operators.

MonetDB interpreter language – MIL

The MonetDB interpreter language is a dynamically typed language that can be used for fast prototyping on top of the MonetDB kernel. MIL enables easy specification of procedures that operate on BATs as well as other MonetDB atomic types. An example of a declaration of a MIL procedure is given below.

```
PROC proc_name(type par_1, type par_2, ..., type par_n) : res_type
```

It contains the name of the procedure (`proc_name`), a set of input parameters ({`par_1,...,par_n`}) of a different `type` (i.e., BAT, integer, real, string), and a type of the result of the execution (`res_type`).

Now we can illustrate how each different implementation of scoring functions is instantiated as a call to one MIL procedure. For example, for score computation, operator $\sqsupset_p$ can be transformed into different calls to MIL procedures that implement different score computation models. Assuming that we implemented language model, Okapi, and tf.idf score computation model (see the following section), these calls can look like:

```
p_containing_t_LMs(bat R_1,bat R_2,flt lambda,int size_type);

p_containing_t_Okapi(bat R_1,bat R_2,flt k_1,flt b,int size_type);

p_containing_t_tfidf(bat R_1,bat R_2,int size_type);
```

Here, `R_1` and `R_2` are BATs of the form `BAT[oid,double]`, `lambda` is a language model parameter λ, and `size_type` parameter determines whether the size is computed as a term count or using region bounds (see Equations 4.2 and 4.3 in the following section). In a BAT *oid* is the region identifier, and *double* value represents the score of a region. `R_1` and `R_2` BATs represent regions in the left and in the right operand respectively.

[1]Although specified in the table, element name, type, and word BATs do not contain string values but integers that encode different strings. Therefore, additional tables exist that store the mapping between these integers and strings they represent for all three BATs (also for the attribute name and value BATs).

Other values of region attributes within the MIL functions can be accessed by forming a join between the R_1 and R_2 BATs and name, type, starting, or end BATs (depicted in Table 4.1) that are persistently stored in a database. The result of the application of such procedures is again `BAT[oid,double]` table, where each region (identified with a unique *oid* identifier) now contains a new score value.

Therefore, each operator at the logical level is instantiated as a call to one MIL procedure. Based on the SRA query plan and the retrieval model specification, the call to the proper MIL functions with specified parameters is achieved. The output of the query execution is a list of elements with their respective scores (`BAT[oid,double]`) reflecting the relevance of an element to a query, obtained after the query execution in the MonetDB kernel. This element list can than be sorted and transformed into the proper output for the evaluation, or used for presenting the results to the end user in textual format.

4.2 Transparent instantiation of retrieval models

In TIJAH (SRA) retrieval models are instantiated following the four elementary retrieval requirements. They are defined using score region algebra abstract functions and abstract operators, but their real implementation is in MIL procedures, as we illustrated in the previous section. For the transparent specification of scoring functions we use auxiliary functions. In these functions $r_i \prec r_j$ is used to denote that the region r_i is contained in the region r_j ($r_i \prec r_j \Leftrightarrow r_j.s < r_i.s \leq r_i.e < r_j.e$). C is used to denote the set of all regions in the collection, i.e., the set of regions formed when the data is translated into the score region algebra data model (see Section 3.2), and *Root* is the collection root node.

The auxiliary functions are used for a number of purposes. The first one counts the number of regions in the region set R, denoted with $|R|$. The second one computes the size of the region r, either based on starting and end index of the region (Equation 4.2) or the number of contained terms (Equation 4.3). Finally, the third one computes the average size of the regions with the region name n in the collection, denoted with $avg\_size(n)$ as depicted in Equation 4.4, also based either on the element index or term count.

$$size^{element}(r) = r.e - r.s - 1 \tag{4.2}$$

$$size^{term}(r) = |\{r_1 \in C | r_1.t = term \wedge r_1 \prec r\}| \tag{4.3}$$

$$avg\_size(n) = \frac{\sum_{r_1 \in C | r_1.n = n} size(r_1)}{|\{r_1 \in C | r_1.n = n\}|} \tag{4.4}$$

The difference between Equation 4.2 and Equation 4.3 is that the former, beside terms, also counts the opening and closing tags and attribute names and values of regions contained in the region r.

4.2.1 Element and term selection

Element and term selection operator $\sigma_{n=name,t=type}(R)$ always selects elements with the computed region score in case $R \neq C$ or with the default score of regions in case $R = C$. This can be seen in Table 3.2. As already mentioned, for the default score for all regions in the collection (C) we choose 1.0.

We use two variants of the $\sigma_{n=name,t=type}(R)$ operator. One has the same notation and the same definition as given in Table 3.2, while the other is denoted as $\sigma^{stem}_{n=name,t=type}(R)$. The latter behaves the same as the former in case $t \neq$ term, while in case $t =$ term it specifies the selection of all the term nodes from the region set R that have the same stem as the word *name* (already stemmed in the query processing at the end-user level). This variant of selection operator is defined as given in Equation 4.5, assuming that $stem(r.n)$ is a function that finds the stem of the word $r.n$.

$$\begin{aligned}\sigma^{stem}_{n=name,t=type}(R) \quad = \quad & \{r \mid r \in R \ \wedge \ ((r.t = \text{term} \ \wedge stem(r.n) \ = name) \\ & \vee \ (r.t \neq \text{term} \wedge r.n = name)) \wedge r.t = type\} \qquad (4.5)\end{aligned}$$

We use only one variant for the specification of element selection: elements are selected by strictly matching their names. For the other variants of element selection on element names, we refer to [139, 145].

4.2.2 Element relevance score computation

Operator $\sqsupset_p$ models element relevance score computation, i.e., the concept that the search elements (regions in the first operand) should contain the term (regions in the second operand). Therefore, the function $f_{\sqsupset}(r_1, R_2)$, applied to a region r_1 and a region set R_2, should result in the numeric value that specifies the relevance of the region (element) r_1 given the (term) regions in R_2 that it contains.

Similarly, operator $\not\sqsupset_p$ models the concept that the search elements (regions in the first operand) should not contain the term (regions in the second operand). Therefore, the function $f_{\not\sqsupset}(r_1, R_2)$, should reflect that the region (element) r_1 has a lower score if it contains the (term) regions from the right operand (R_2).

Many retrieval models exist that specify how relevant is a document (element) given the terms it contain (see Section 2.1). We choose to adapt five of these models to implement score computation aspect. Two most important reasons for this choice are that these models are state-of-the-art retrieval models and that we use them later in our experiments in Chapters 5 to 7. The five retrieval models we use to specify score computation are: Boolean, tf.idf [193], Language Models (LMs) [95], the Okapi (INQUERY) model [30, 181], and the Garden Point XML (GPX) model [77][2].

The simple Boolean formula for score computation ($f_{\sqsupset}$) is given by:

$$f_{\sqsupset}^{\text{Bool}}(r_1, R_2) = r_1.p \cdot sgn(|\{r_2 \in R_2 | r_2 \prec r_1\}|) \tag{4.6}$$

The tf.idf score computation formula is based on the Equation 2.5 given in Section 2.1.1. Its adaptation in SRA is as follows:

$$f_{\sqsupset}^{\text{tf.idf}}(r_1, R_2) = r_1.p \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot ln \frac{|\{r \in C | r.n = r_1.n\}|}{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}|} \tag{4.7}$$

The language model score computation can be instantiated based on auxiliary functions, and the basic model given in Equation 2.14 (Section 2.1.1) as:

$$f_{\sqsupset}^{\text{LMs}}(r_1, R_2) = r_1.p \cdot \left(\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{size(r_1)} + (1 - \lambda) \frac{|R_2|}{size(Root)} \right) \tag{4.8}$$

The Okapi formula is derived from the original model (described by Equation 2.8 in Section 2.1.1) by removing the third factor from the product. The reason is that the third factor is based on a size of the query and is not supported in other models. The complex function $f_{\sqsupset}^{\text{Okapi}}$ is specified as:

$$\begin{aligned} f_{\sqsupset}^{\text{Okapi}}(r_1, R_2) &= r_1.p \\ &\cdot ln \frac{|\{r \in C | r.n = r_1.n\}| - |\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}| + 0.5}{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}| + 0.5} \\ &\cdot \frac{(k_1 + 1) \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{k_1((1 - b) + b \frac{size(r_1)}{avg\_size(r_1.n)}) + \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p} \end{aligned} \tag{4.9}$$

For the element relevance score computation in the GPX model we use the specification given in [77] and repeated in Equation 2.20 in Section 2.1.2 of this thesis:

$$f_{\sqsupset}^{\text{GPX}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{|R_2|} \tag{4.10}$$

[2]Although this is not a well known retrieval model we have chosen it as it is among the most effective ones for XML retrieval presented at INEX 2004 workshop [74].

Although this is a slightly different formula than the original GPX model (due to the specification of region algebra operators) it showed equally good performance in our previous experiments [137].

Similarly to modeling positive score computation aspect we model negative score computation aspect. While for Boolean and language models the $f_{\not\sqsupset}$ formula can be easily derived, this is not the case for the other three score computation models. The reason is that the result of Boolean and language model score computation functions is normalized, i.e., the resulting score is in the range $[0, 1]$. The $f_{\not\sqsupset}$ formulas for this two cases are given in Equations 4.11 and 4.12.

$$f_{\not\sqsupset}^{\text{Bool}}(r_1, R_2) = r_1.p \cdot (1 - sgn(|\{r_2 \in R_2 | r_2 \prec r_1\}|)) \tag{4.11}$$

$$f_{\not\sqsupset}^{\text{LMs}}(r_1, R_2) = r_1.p \cdot \left(1 - \left(\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{size(r_1)} + (1-\lambda)\frac{|R_2|}{size(Root)}\right)\right) \tag{4.12}$$

For the tf.idf and Okapi we normalized the scores. If we denote the products on the right side of the Equations 4.7 and 4.9 after the $r_1.p$ with $g_{\sqsupset}$, and its normalization in the range $[0, 1]$ with $\overline{g_{\sqsupset}}$ we can define tf.idf and Okapi score computations as depicted in Equations 4.13 to 4.16. For example, the values are transformed (normalized) from the range $[1 \cdot ln1, |\{r \in C | r.t = \text{term}\}| \cdot |\{r \in C | r.t = \text{node}\}|]$ to $[0, 1]$ in case of $\overline{g_{\sqsupset}^{\text{tf.idf}}}$ function.

$$f_{\sqsupset}^{\text{tf.idf}}(r_1, R_2) = r_1.p \cdot \overline{g_{\sqsupset}^{\text{tf.idf}}} \tag{4.13}$$

$$f_{\not\sqsupset}^{\text{tf.idf}}(r_1, R_2) = r_1.p \cdot \left(1 - \overline{g_{\sqsupset}^{\text{tf.idf}}}\right) \tag{4.14}$$

$$f_{\sqsupset}^{\text{Okapi}}(r_1, R_2) = r_1.p \cdot \overline{g_{\sqsupset}^{\text{Okapi}}} \tag{4.15}$$

$$f_{\not\sqsupset}^{\text{Okapi}}(r_1, R_2) = r_1.p \cdot \left(1 - \overline{g_{\sqsupset}^{\text{Okapi}}}\right) \tag{4.16}$$

However, for the GPX formula we had to apply different kind of normalization. The aim of the formulas $f_{\sqsupset}^{\text{GPX}}$ and $f_{\not\sqsupset}^{\text{GPX}}$ is to produce zero scores for element regions that do not contain term regions for $\sqsupset_p$, and element regions that contain term regions for $\not\sqsupset_p$. Therefore, the definition of $f_{\not\sqsupset}^{\text{GPX}}$ function is the same as the definition of Boolean $f_{\not\sqsupset}^{\text{Bool}}$ function given in Equation 4.11.

4.2.3 Relevance score combination

The abstract operator $\otimes$ specifies how scores are combined in an AND expression, denoted in SRA by $\sqcap_p$, while the operator $\oplus$ defines score combination in an OR expression, denoted in SRA with $\sqcup_p$. We make different choices for the implementation of abstract score combination operators. These choices are based on the most frequently used operations for combining scores in traditional flat text retrieval systems described in Section 2.1.

We choose some simple implementations for AND and OR score combination such as sum, product, minimum, and maximum, i.e., $\{\oplus, \otimes\} := \{+, *, min, max\}$. Additionally, following [30] and [77], we also define the two abstract operators as probabilistic sum shown in Equations 4.17 and as exponential sum shown in Equation 4.18:

$$r_1.p\{\oplus^{prob}, \otimes^{prob}\}r_2.p = 1 - (1 - r_1.p) \cdot (1 - r_2.p) \tag{4.17}$$

$$r_1.p\{\oplus^{exp}, \otimes^{exp}\}r_2.p = \begin{cases} r_1.p + r_2.p & \text{if } r_1.p = 0 \vee r_2.p = 0 \\ \\ A \cdot (r_1.p + r_2.p) & \text{otherwise} \end{cases} \tag{4.18}$$

Both score combination operators can be implemented using the same definition of abstract scoring function. This is due to the fact that in most cases users mix AND and OR score combination when they specify their requests and as this aspect is not thoroughly examined in the structured retrieval process. Recall that in structured retrieval the score combination aspect models also the combination of scores of elements that do not directly contain the terms. All these variants of score combination implementations are tested in Chapters 5 and 6.

4.2.4 Relevance score propagation

The operator $\blacktriangleright$ specifies propagation of scores to the containing elements, e.g., from 'image' to 'story' elements. Thus, the function $f_{\blacktriangleright}(r_1, R_2)$ specifies upwards element score propagation. Among many possibilities we choose the following four options:

- an average of region scores from the right operand contained in the region from the left operand, multiplied by the left operand region score (Equation 4.19)
- a simple sum of region scores from the right operand contained in the region from the left operand, multiplied by the left operand region score (Equation 4.20)

- a weighted sum of region scores from the right operand contained in the region from the left operand, multiplied by the left operand region score and normalized by the left operand region size (Equation 4.21)
- a weighted sum of region scores in the right operand contained in the region from the left operand, multiplied by the left operand region score and normalized by the sum of all sizes of the contained right operand regions (Equation 4.22).

$$f_{\blacktriangleright}^{\text{avg}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{|r_2 \in R_2 | r_2 \prec r_1|} \tag{4.19}$$

$$f_{\blacktriangleright}^{\text{sum}}(r_1, R_2) = r_1.p \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \tag{4.20}$$

$$f_{\blacktriangleright}^{\text{wsuma}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot size(r_2)}{size(r_1)} \tag{4.21}$$

$$f_{\blacktriangleright}^{\text{wsumd}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot size(r_2)}{\sum_{r_2 \in R_2 | r_2 \prec r_1} size(r_2)} \tag{4.22}$$

We also perform smoothing for the score propagation to avoid zero scores for regions from the left operand that do not contain regions from the right operand, e.g., 'story' elements that do not contain 'figure' elements. The score propagation smoothing uses the frequency of regions (elements) from the right operand contained in regions (elements) having the name $r_1.n$ (name of the region in the left operand). The influence of this "element frequency" is regulated by a smoothing parameter ω. If we denote the right factor in the product in Equations 4.19 to 4.22 with $g_{\blacktriangleright}(r_1, R_2)$, we can express the score propagation with structured smoothing as depicted in Equation 4.23.

$$\begin{aligned} f_{\blacktriangleright}^{\text{smooth}}(r_1, R_2) &= r_1.p \cdot (\omega \cdot g_{\blacktriangleright}(r_1, R_2) + \\ &+ (1-\omega) \cdot \frac{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}|}{|\{r \in C | r.n = r_1.n\}|}) \end{aligned} \tag{4.23}$$

On the other hand, to model downwards score propagation, e.g., the propagation of scores from 'story' to 'image' elements, operator $\blacktriangleleft$ is defined with the abstract function $f_{\blacktriangleleft}(r_1, R_2)$. Similarly to $f_{\blacktriangleright}(r_1, R_2)$, $f_{\blacktriangleleft}(r_1, R_2)$ can be implemented in different ways. However, we have chosen the same four propagation functions as for the upwards score propagation and adapted it to downwards score propagation. The functions are presented in Equations 4.24 to 4.27.

$$f_{\blacktriangleleft}^{\text{avg}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_1 \prec r_2} r_2.p}{|r_2 \in R_2 | r_1 \prec r_2|} \tag{4.24}$$

$$f_{\blacktriangleleft}^{\text{sum}}(r_1, R_2) = r_1.p \cdot \sum_{r_2 \in R_2 | r_1 \prec r_2} r_2.p \tag{4.25}$$

$$f_{\blacktriangleleft}^{\text{wsuma}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_1 \prec r_2} r_2.p \cdot size(r_2)}{size(r_1)} \tag{4.26}$$

$$f_{\blacktriangleleft}^{\text{wsumd}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_1 \prec r_2} r_2.p \cdot size(r_2)}{\sum_{r_2 \in R_2 | r_1 \prec r_2} size(r_2)} \tag{4.27}$$

Similar to upwards score propagation, we introduce the smoothed version of downwards score propagation functions. If we denote the right factor in the Equations 4.24 to 4.27 with $g_{\blacktriangleleft}(r_1, R_2)$ we can express the downwards score propagation with structured smoothing as depicted in Equation 4.28.

$$\begin{aligned} f_{\blacktriangleleft}^{\text{smooth}}(r_1, R_2) &= r_1.p \cdot (\omega \cdot g_{\blacktriangleleft}(r_1, R_2) + \\ &\quad + (1 - \omega) \cdot \frac{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_1 \prec r_2\}|}{|\{r \in C | r.n = r_1.n\}|}) \end{aligned} \tag{4.28}$$

In the specification of score propagation function we follow the pioneering work on augmentation weights for propagating scores from child elements to their parents by Fuhr and Großjohann [71] and Grabs and Schek [81]. They used manually selected predefined downweighting factors for this propagation (see Section 2.1.2). Later, many approaches used the same concept except that they used more information in implementing upwards score propagation, such as the size of elements [154] or nesting level [194]. We use element sizes to dynamically determine the augmentation weights when propagating scores upwards and downwards. However, other implementations are possible in SRA (see also Section 7.1 in this thesis).

4.3 SRA operator properties

The goal of studying operator properties is to enable logical query optimization, such as relational algebra query optimization, as can be seen in, e.g., [16], and obtaining a gain in efficiency. However, as the efficiency is not the main issue in this thesis we only illustrate the potential usage of SRA operator properties for query optimization and also leave the discussion on numeric stability problems for future research.

Our study on score region algebra shows that there are only few operators that have the basic operator properties such as: identity, inverse, commutativity, associativity, and distributivity. However, there is a number of region algebra specific properties which can be considered as a special case of distributivity and associativity properties. We mostly focus on such properties. For the more elaborate analysis of other properties, see the overview given in [140].

4.3.1 Special properties of selection operators

One interesting and potentially useful property for query optimization is presented in Properties 4.29 to 4.32. It states that the sequence of performing containment selection operations can be changed.

$$(R_1 \sqsupset R_2) \sqsupset R_3 = (R_1 \sqsupset R_3) \sqsupset R_2 \tag{4.29}$$

$$(R_1 \sqsubset R_2) \sqsubset R_3 = (R_1 \sqsubset R_3) \sqsubset R_2 \tag{4.30}$$

$$(R_1 \sqsupset R_2) \sqsubset R_3 = (R_1 \sqsubset R_3) \sqsupset R_2 \tag{4.31}$$

$$(R_1 \sqsubset R_2) \sqsupset R_3 = (R_1 \sqsupset R_3) \sqsubset R_2 \tag{4.32}$$

These properties can be useful if we know that some of the operations are more selective than the others. For example, assume that we have a collection of stories where only few of them are enclosed in 'short_story' tags, and that all stories contain a prolog where most prologs include some images. Then, the following query expression can be issued `//short_story//prolog[.//image]`. It searches for 'prolog' elements in a 'short_story' element containing 'image' elements. This query can be expressed in SRA as:

$$(R^n_{\text{prolog}} \sqsupset R^n_{\text{image}}) \sqsubset R^n_{\text{short\_story}}$$

Here, all the prologs (in all the 'story' elements) in the collection that contain images would be selected, and than only the ones that are inside the 'short_story' elements would be filtered. It would be more efficient to first select prologs that are in a few 'short_story' elements, and than to select those that contain images. Therefore, Property 4.31 can be used to express this query in the following way:

$$(R^n_{\text{prolog}} \sqsubset R^n_{\text{short\_story}}) \sqsupset R^n_{\text{image}}$$

Another interesting property is that the name and type selection operator can be pushed through the containment expression, as depicted in Property 4.33 and 4.34.

$$\sigma_{n=name,t=type}(R_1 \sqsupset R_2) = \sigma_{n=name,t=type}(R_1) \sqsupset R_2 \tag{4.33}$$

$$\sigma_{n=name,t=type}(R_1 \sqsubset R_2) = \sigma_{n=name,t=type}(R_1) \sqsubset R_2 \tag{4.34}$$

The usage of such properties can be illustrated on the following example. The NEXI expression `image//*` selects all element nodes in the collection (denoted with '`*`') that are contained in an 'image' element. In SRA this can be expressed as follows.

$$\sigma_{t=\text{node}}(C) \sqsubset \sigma_{n=\text{image},t=\text{node}}(C)$$

Knowing that not many 'image' elements exist in the collection it might be more efficient to express this need in SRA as:

$$\sigma_{t=\text{node}}(C \sqsubset \sigma_{n=\text{image},t=\text{node}}(C))$$

The latter expression first selects the regions that are contained in the 'image' region (assuming there are only few) and test their type with the $\sigma_{t=\text{node}}$ operator. This should be less memory consuming and faster then first selecting all the element nodes in the collection and then testing them on containment.

4.3.2 Properties of score manipulation operators

Considering the properties of scoring operators (operators that include score computation, combination, and propagation) we can exert that some of the properties follow the ones defined for the region algebra that do not include score manipulation (see [38] and [140] for discussions on region algebra operator properties). Some of them hold only if some conditions are satisfied (conditional properties which depend on the underlying retrieval model), while some of them are no longer valid.

Score combination operators

Operator $\sqcap_p$ defines the Boolean-like AND combination of scores obtained for two regions with the same region bounds (i.e., values of s and e attributes). It has the identity and inverse element properties, but only in case the default score value for all regions in the initial region set is the value which is the identity element for the abstract operator $\otimes$. The inverse element is than the set of all regions in the collection C, as depicted in Property 4.35. For example, this is true for the default region score of 1 and for the $\otimes$ operator implemented as minimum[3] or product: $min(r.p, 1) = min(1, r.p) = r.p$, $\forall r \in R$ and $r.p \cdot 1 = 1 \cdot r.p = r.p$, $\forall r \in R$.

$$R \sqcap_p C = C \sqcap_p R = R \tag{4.35}$$

Similarly, if the default score for regions in the collection is 0, then Property 4.35 would hold for $\otimes$ implemented as maximum, probabilistic sum (Equation 4.17), sum, and exponential sum (Equation 4.18). For example in case of $\otimes^{prob}$ it would be $1-(1-r.p)\cdot(1-0) = 1-(1-0)\cdot(1-r.p) = r.p$, $\forall r \in R$.

Furthermore, the operator $\sqcap_p$ is commutative or associative if the operator $\otimes$ is commutative (Property 4.36) or associative (Property 4.37) respectively. The commutativity holds for all six implementations of $\otimes$ abstract operator given in Section 4.2, while the associativity is true only for $\otimes$ implemented as maximum, minimum, product, and sum.

$$R_1 \sqcap_p R_2 = R_2 \sqcap_p R_1 \tag{4.36}$$

$$(R_1 \sqcap_p R_2) \sqcap_p R_3 = R_1 \sqcap_p (R_2 \sqcap_p R_3) \tag{4.37}$$

An extension of the set union operator is given by the $\sqcup_p$ operator. It defines the Boolean-like OR combination of scores for two regions. Similarly to $\sqcap_p$ operator, operator $\sqcup_p$ has the identity and inverse element properties (Property 4.38), but for all implementations of $\oplus$ operator as the inverse region set for the $\sqcup_p$ operator is an empty set ($\emptyset$).

$$R \sqcup_p \emptyset = \emptyset \sqcup_p R = R \tag{4.38}$$

As in the $\sqcap_p$ operator case, commutativity and associativity properties depend on the definition of $\oplus$ operator. In other words, operator $\sqcup_p$ is commutative or associative if the operator $\oplus$ is commutative or associative respectively. As we used the same implementation for the $\oplus$ operator as for the $\otimes$ operator, Property 4.39 is

[3]Assuming that the scoring functions are normalized in the algebra, i.e., that the resulting score of the abstract functions and operators is in the range $[0, 1]$.

true for all six implementations of abstract OR score combination operator, while Property 4.40 is true only for $\oplus$ implemented as maximum, minimum, product, or sum.

$$R_1 \sqcup_p R_2 = R_2 \sqcup_p R_1 \tag{4.39}$$

$$(R_1 \sqcup_p R_2) \sqcup_p R_3 = R_1 \sqcup_p (R_2 \sqcup_p R_3) \tag{4.40}$$

Furthermore, in some cases the AND score combination operator distributes over the OR score combination operator (Property 4.41) and vice versa (Property 4.42). $\sqcap_p$ distributes over $\sqcup_p$ in the following two cases: (1) $\otimes$ and $\oplus$ implemented as maximum and minimum and (2) $\otimes$ implemented as product and $\oplus$ implemented as sum. Similarly, $\sqcup_p$ distributes over $\sqcap_p$ in the following two cases: (1) $\otimes$ and $\oplus$ implemented as maximum and minimum and (2) $\oplus$ implemented as product and $\otimes$ implemented as sum. This is due to distributivity of minimum and maximum operators on real numbers, and distributivity of product with respect to sum on real numbers.

$$R_1 \sqcap_p (R_2 \sqcup_p R_3) = (R_1 \sqcap_p R_2) \sqcup_p (R_1 \sqcap_p R_3) \tag{4.41}$$

$$R_1 \sqcup_p (R_2 \sqcap_p R_3) = (R_1 \sqcup_p R_2) \sqcap_p (R_1 \sqcup_p R_3) \tag{4.42}$$

Score computation and score propagation operators

Similarly to Properties 4.29 to 4.32, a combination of one scoring and one containment selection operator holds. If we denote containment selection operators, i.e., $\{\sqsupset, \sqsubset\}$, with *op* and scoring operators, i.e., $\{\sqsupset_p, \not\sqsupset_p, \blacktriangleright, \blacktriangleleft\}$, with op_p, these properties can be expressed as shown in Properties 4.43 and 4.44.

$$(R_1\ op\ R_2)\ op_p\ R_3 = (R_1\ op_p\ R_3)\ op\ R_2 \tag{4.43}$$

$$(R_1\ op_p\ R_2)\ op\ R_3 = (R_1\ op\ R_3)\ op_p\ R_2 \tag{4.44}$$

Consider the following NEXI query `//prolog//p[about(., garden)]` for example. Using Property 4.44, the query can be expressed in SRA in two ways as shown below.

$$(R^n_{\text{p}} \sqsupset_p R^w_{\text{garden}}) \sqsubset R^n_{\text{prolog}} = (R^n_{\text{p}} \sqsubset R^n_{\text{prolog}}) \sqsupset_p R^w_{\text{garden}}$$

The expression on the left side of the equation is likely to be less efficient. It states that the relevance score should be computed for all paragraphs in the collection with respect to contained 'garden' term regions, and then the paragraphs from prologs should be selected. The expression on the right side first selects only the paragraphs inside 'prolog' elements and than computes the relevance score of those paragraphs with respect to 'garden' term regions.

The selection operator on region name and type (σ) can be pushed through the expressions that involve score manipulation among two regions, similarly as it is the case for containment operators (see Properties 4.33 and 4.34). This is depicted in Property 4.45.

$$\sigma_{n=name,t=type}(R_1\ op_p\ R_2) = \sigma_{n=name,t=type}(R_1)\ op_p\ R_2 \tag{4.45}$$

On the other hand, properties 4.29 to 4.32, in case both operators include score manipulation, do not hold in general. However, for some special retrieval model instantiations they hold. Looking at the specification of our score computation ($f_{\sqsupset}(r_1, R_2)$, and $f_{\not\sqsupset}(r_1, R_2)$) and score propagation ($f_{\blacktriangleright}(r_1, R_2)$ and $f_{\blacktriangleleft}(r_1, R_2)$) functions we can see that they can be expressed in the following way $f(r_1, R_2) = r_1.p \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_2)$. We use f and g, to replace $f_{\sqsupset}$, $f_{\not\sqsupset}$, $f_{\blacktriangleright}$, $f_{\blacktriangleleft}$, and $g_{\sqsupset}$, $g_{\not\sqsupset}$, $g_{\blacktriangleright}$, and $g_{\blacktriangleleft}$, respectively. Based on the parameter list we can see that the function $g(r_1.s, r_1.e, r_1.n, r_1.t, R_2)$ does not depend on the score of the region r_1. In such case the Property 4.46 holds, where $op1_p \in \{\sqsupset_p, \not\sqsupset_p, \blacktriangleright, \blacktriangleleft\}$ and $op2_p \in \{\sqsupset_p, \not\sqsupset_p, \blacktriangleright, \blacktriangleleft\}$.

$$(R_1\ op1_p\ R_2)\ op2_p\ R_3 = (R_1\ op2_p\ R_3)\ op1_p\ R_2 \tag{4.46}$$

In this case, the score for each region in the result region set, denoted with $r.p$, is computed as:

$$\begin{aligned} r.p &= (r_1.p \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_2)) \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_3) \\ &= (r_1.p \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_3)) \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_2) \end{aligned}$$

Furthermore, if the score value for all regions in the first operand R_1 is equal to 1 (default value for all regions), and abstract AND or OR score computation operators are implemented as product, the following properties hold:

$$(R_1\ op1_p\ R_2)\ op2_p\ R_3 = (R_1\ op1_p\ R_2) \sqcap_p (R_1\ op2_p\ R_3) \tag{4.47}$$

$$(R_1\ op1_p\ R_2)\ op2_p\ R_3 = (R_1\ op1_p\ R_2) \sqcup_p (R_1\ op2_p\ R_3) \tag{4.48}$$

i.e., for every region in the result set we obtain region score $r.p$:

$$\begin{aligned} r.p &= (1 \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_2)) \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_3) \\ &= (1 \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_2)) \cdot (1 \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_3)) \end{aligned}$$

The Property 4.47 (Property 4.48) can be used for NEXI expressions that look like `//image[about(., beautiful flower)]` (`//image[about(., beautiful) or about(., flower)]`). The SRA counterpart of such NEXI query can be simplified using Property 4.47 as shown below:

$$(R^n_{\text{image}} \sqsupset_p R^w_{\text{beautiful}}) \sqcap_p (R^n_{\text{image}} \sqsupset_p R^w_{\text{flower}}) = (R^n_{\text{image}} \sqsupset_p R^w_{\text{beautiful}}) \sqsupset_p R^w_{\text{flower}}$$

Under the same condition as for Properties 4.47 and 4.48, except that the abstract AND and OR score computation operators are implemented as sum, other two special properties hold: Property 4.49 and Property 4.50.

$$(R_1 \sqcap_p R_2)\ op_p\ R_3 = (R_1\ op_p\ R_3) \sqcap_p (R_2\ op_p\ R_3) \tag{4.49}$$

$$(R_1 \sqcup_p R_2)\ op_p\ R_3 = (R_1\ op_p\ R_3) \sqcup_p (R_2\ op_p\ R_3) \tag{4.50}$$

i.e.,

$$\begin{aligned} r.p &= (r_1.p + r_2.p) \cdot g(r_{1,2}.s, r_{1,2}.e, r_{1,2}.n, r_{1,2}.t, R_3) \\ &= (r_1.p \cdot g(r_1.s, r_1.e, r_1.n, r_1.t, R_3)) + (r_2.p \cdot g(r_2.s, r_2.e, r_2.n, r_2.t, R_3)) \end{aligned}$$

where $r_{1,2}$ denotes the regions that are either in one of the region sets R_1 and R_2 in case of Property 4.50, or in both of them in case of Property 4.49.

The following NEXI query `//(image|video)[about(., garden)]` can be used for illustrating Property 4.50. A possible SRA expression for such a NEXI query is shown below. This property can be useful if we have parallel processing at the physical level for query parts. In that case one process can compute the scores of the first subexpression ($R^n_{\text{image}} \sqsupset_p R^w_{\text{garden}}$) and the other of the second subexpression ($R^n_{\text{video}} \sqsupset_p R^w_{\text{garden}}$) and then they can be combined.

$$(R^n_{\text{image}} \sqcup_p R^n_{\text{video}}) \sqsupset_p R^w_{\text{garden}} = (R^n_{\text{image}} \sqsupset_p R^w_{\text{garden}}) \sqcup_p (R^n_{\text{video}} \sqsupset_p R^w_{\text{garden}})$$

There are many properties that can be identified in score region algebra when combining different instantiations of scoring functions. In this section we pointed out that the SRA framework can be used for query rewriting and optimization by explaining some interesting properties. However, as the analysis of the SRA properties is not the main topic of this thesis, further discussion is left for future work.

4.4 Summary

The architecture and the components of our three-level database system for structured information retrieval called TIJAH are the main topic of the first part of this chapter. The components of the end-user, logical, and physical levels are presented and their functionality is explained. The emphasis is on transparent specification of retrieval models along all three levels. A special attention is given to the retrieval dictionary where the data for controlling the end-user to logical and logical to physical mappings are stored.

The central part of the chapter is dedicated to the instantiation of various retrieval models in score region algebra. Different implementations of scoring functions in SRA that follow the elementary structured retrieval requirements are presented. These implementations are specified using abstract functions and abstract operators withing the SRA selection, score computation, score combination, and score propagation operators. For each operator we explained several possible instantiations and justify our choice. We based this choices on state-of-the-art retrieval models and approaches for structure retrieval developed in the last few years.

Finally, some properties of algebra operators are discussed at the end. The properties are discussed only to emphasize the potential of using them for score region algebra logical query optimization. The full analysis of the SRA operator properties is a research topic on its own and is left for future research.

Chapter 5

Component Retrieval

To test our three-level database system for structured IR and to learn more about structured retrieval, several series of experiments are evaluated in this chapter. The experiments are performed on the INitiative for the Evaluation of XML Retrieval (INEX) [74] test collection. The chapter starts with the aim of the presented experiments, described through several hypotheses. It then presents the INEX test collection and the experimental setup. The chapter continues with analyzing different implementations of score computation, score combination, and score propagation functions. It ends with a short overview of the experimental results, and guidelines for further studies.

This chapter is partially based on papers published (1) in the Proceedings of the 14^{th} ACM International Conference on Information Knowledge and Management (CIKM) [138] and (2) as a Centre for Telematics and Information Technology Technical Report [141].

5.1 Motivation

The experimentation presented in this chapter is motivated by the following three goals:

1. to provide guidelines for implementing score computation, combination, and propagation functions, by strictly following structured constraints
2. to point out the main differences in modeling structured IR with respect to modeling flat text IR
3. to emphasize the benefits of transparent instantiation of retrieval models in the score region algebra.

The queries that we use for the experimentation are content-and-structure (CAS) queries as defined in INEX [74]. Having a transparent structured retrieval framework, and having such a structured collection, we design distinct experimental series. We design these series to test different aspects of structured information retrieval. They are explained below, by introducing six hypotheses. The hypotheses are based on flat text (TREC [216]) experimental results and structured retrieval experiments presented in INEX [74]. They are tested on structured retrieval tasks in later sections in this chapter.

5.1.1 How to compute scores?

The experimentation starts with checking what is the best way to compute the element score in structured retrieval. For these experiments we use different score computation models (see Section 4.2.2). Following the experimental results when using state-of-the-art retrieval models we can pose the first hypothesis.

Hypothesis 1. The effectiveness of more advanced retrieval models, i.e., GPX, language models, and Okapi, should be higher than the effectiveness of simple retrieval models, such as Boolean and tf.idf, on structured retrieval (see Section 2.1).

Additionally, in these series of experiments we test two other aspects of score computation. The first one tries to determine which of the element size computation formulas, i.e., using region boundaries (Equation 4.2) or using term count (Equation 4.3), gives higher effectiveness. The second tries to determine whether stemming helps in structured retrieval. Therefore, the next hypotheses are as follows.

Hypothesis 2. Following the flat text IR approaches, we expect that in structured retrieval it is more effective to compute the sizes of elements (documents) by counting the number of terms contained in them than using region bounds.

Hypothesis 3. Following flat text IR approaches, the usage of stemming should result in higher effectiveness also in structured retrieval.

Another component of the element relevance score computation is tested in the experiments. The goal is to find out whether the values of parameters for different retrieval models are the same as parameter values for state-of-the-art retrieval models. Also, we investigate for which score computation models length prior (see Section 5.3) helps.

5.1.2 How to combine scores?

In this chapter we analyze different implementations of AND and OR score combination functions, in composition with different score computation models. The goal is to test how distinct score computations behave when different implementations of score combination are employed. Hypotheses are as follows.

Hypothesis 4. Following the specification of different retrieval models given in Section 2.1, the AND score combination implemented as: (1) exponential sum for GPX, (2) product for language model, (3) sum for Okapi, and (4) sum for tf.idf score computation model should give higher effectiveness than other implementations.

Hypothesis 5. Following the specification of different retrieval models given in Section 2.1, the OR score combination implemented as: (1) exponential sum for GPX, (2) sum for language model, (3) sum for Okapi, and (4) sum for tf.idf score computation model should give higher effectiveness than other implementations.

The results of testing these hypothesis would answer the question whether the specification of state-of-the-art retrieval models holds for structured retrieval. The best composition of score computation and score combination implementations are then used for further experiments with the score propagation.

Furthermore, besides comparing the effectiveness of different compositions of score computation and score combination implementations, these compositions are analyzed per query. The goal is to detect relations between retrieval models and queries that have different complexity.

5.1.3 How to propagate scores?

The effectiveness of retrieval models when using different implementations of score propagation functions are also analyzed. As this requirement is specific to structured retrieval, not many experiments are presented in the literature that point out what would be more effective implementation. The implementations usually model child-parent upwards score propagation where scores are computed as a weighted sum of child element scores [71, 81, 154], or descendant-ancestor upwards propagation using the distance between element levels in the hierarchical document structure [194]. However, what is the best way for downweighting scores when propagating them upwards and downwards is not clear. The only hint is that some form of weighted sum should be used. Therefore, we make one hypothesis with respect to score propagation, considering the four implementations of score propagation functions in Section 4.2.4 (Equations 4.19 to 4.22).

Hypothesis 6. Based on the state-of-the-art structured retrieval models we expect that the weighted sum approaches are more appropriate for modeling score propagation than simple sum or average.

Besides analyzing the effectiveness of different score propagation functions we also discuss the influence of structured smoothing on retrieval results.

Having discussed the experimental results of composing score computation, score combination, and score propagation, we can point out what are the important features that need to be considered when specifying retrieval models for structured IR. This is addressed in Section 5.7.

5.2 Structured IR evaluation: INEX

Retrieval evaluation initiatives provide test collections consisting of a *document collection*, a set of *user requests* (*topics*), a basic evaluation criterion used for *relevance assessments*, and *evaluation metrics* [180]. In order to setup an evaluation initiative the *objective* of the evaluation must be specified. The most common objective in IR evaluation initiatives is retrieval system effectiveness. Based on the objective, measure (metric) development can follow. Metric compares the system output, obtained by executing user requests over document collection, to the relevance assessments.

The evaluation process can be defined as a process that compares the system output, i.e., ranked result list of relevant components with relevance values assigned to them, with the assessments output using the evaluation metric. The result of this comparison should show the performance of a system (system's effectiveness), usually depicted as a recall-precision graph [12]. A recall-precision graph is a graph where precision is reported for different recall points, e.g., 0.1, 0.2, ..., 1.0 (see Section 5.2.4 for the definition of recall and precision).

The INitiative for the Evaluation of XML Retrieval (INEX) [74] started in 2002 as a yearly evaluation effort aimed at providing an infrastructure and a framework for evaluating the performance of structured retrieval systems. INEX was inspired largely by work on laboratory-style evaluation of information retrieval systems developed in TREC [216]. INEX measures the effectiveness of the access to content that is structured using extensible markup language (XML). As such, INEX provides a large XML data collection and appropriate methods for the evaluation of content-oriented XML retrieval systems [74]. Among many tasks at INEX, we focus on the INEX Ad-hoc task.

The objective of the INEX Ad-hoc evaluation is structured IR effectiveness. It is defined as the system's ability to retrieve the most specific relevant elements, which are exhaustive to the topic of request. Here, exhaustivity describes the extent to which a retrieved component discusses the topic of request, while specificity describes the extent to which the retrieved component focuses on the topic of request.

5.2.1 Data collection

For our experiments we use the INEX 2003 and 2004 data collection consisting of 494MB of data. The data collection consists of 12,107 articles of the IEEE Computer Society's publications from 12 magazines and 6 transactions. It covers the period of 1995–2002. The data collection has a complex XML structure and contains scientific articles of varying length. On average an article contains 1,532 XML nodes and the average depth of a node is 6.9.

An excerpt from the Document Type Definition (DTD) for the IEEE collection of articles is given in Figure 5.1. It shows that the collection consists of books containing journals which contain articles. Usually, articles are formed out of a

Figure 5.1: Document Type Definition (DTD) for the INEX IEEE XML collection.

```
<-- base elements -->
<!ELEMENT books       (journal*)>
<!ELEMENT journal     (title, issue, publisher, graphicc?, (sec1|article|sbt)*)>
<!ELEMENT sec1        (title)>
<!ELEMENT title       (%parmat;)*>
<!ELEMENT issue       (#PCDATA)>
<!ELEMENT publisher   (#PCDATA)>
<!ELEMENT graphicc    (#PCDATA)>
<!ATTLIST graphicc    filename CDATA       #REQUIRED>
<!ELEMENT article     (fno, doi?, fm, bdy, bm?)>

<!ELEMENT fno         (#PCDATA)>    <!-- article ID (no entity references) -->
<!ATTLIST fno         fid      NMTOKEN     #IMPLIED>

<!ELEMENT doi         (#PCDATA)>

...

<-- body matter -->
<!ELEMENT bdy         (%secmat;|sec)*>

<!-- section -->
<!ELEMENT sec         (no?, (sbt|st|%secmat;)*, ss1*, ss2*, ss3*)>
<!ATTLIST sec         type     CDATA       #IMPLIED>

<!ELEMENT ss1         (no?, (st|sbt|%secmat;)*, ss2*)>
<!ELEMENT ss2         (no?, (st, sbt*)?, (%secmat;)*, ss3*)>
<!ELEMENT ss3         (no?, (st, sbt*)?, (%secmat;)*)>

<!ELEMENT brief       (p*)>

<!ELEMENT dialog      (question|answer)*>
<!ELEMENT question    (p*)>
<!ELEMENT answer      (p|list)*>

...

<!-- paragraphs -->
<!ELEMENT ilrj        (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT ip1         (%parmat;|%secparmat;|h3|h4)*>
<!ATTLIST ip1         id       ID          #IMPLIED>
<!ELEMENT ip2         (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT ip3         (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT ip4         (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT ip5         (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT item-none   (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT p           (%parmat;|%secparmat;|h3|h4)*>
<!ATTLIST p           id       ID          #IMPLIED
                      ind      (hang|after|none) "none"
                      align    (left|right|center) "left">
<!ELEMENT p1          (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT p2          (%parmat;|%secparmat;|h3|h4)*>
<!ELEMENT p3          (%parmat;|%secparmat;|h3|h4)*>

...
```

Figure 5.2: An example topic specification: INEX 2004 CAS topic 127.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">

<inex_topic topic_id="127" query_type="CAS" ct_no="13">

  <title>//sec//(p|fgc)[about( ., Godel Lukasiewicz and other fuzzy
  implication definitions)]</title>

  <description>Find paragraphs or figure-captions containing the definition
  of Godel, Lukasiewicz or other fuzzy-logic implications</description>

  <narrative>Any relevant element of a section must contain the definition
  of a fuzzy-logic implication operator or a pointer to the element of the
  same article where the definition can be found. Elements containing
  criteria for identifying or comparing fuzzy implications are also of
  interest. Elements which discuss or introduce non-implication fuzzy
  operators are not relevant. </narrative>

  <keywords>Godel implication, Lukasiewicz implication, fuzzy implications,
  fuzzy-logic implication </keywords>

</inex_topic>
```

front matter (`fm`), body (`bdy`), and back matter (`bm`). Body contains a number of sections (`sec`, `ss1`, `ss2`, etc.) and sections contain paragraphs (`p`, `ip1`, `ip2`, etc.), lists, etc. On documents, having such structure, diverse set of structured queries can be expressed as explained below.

5.2.2 Topic sets

The topics are created by the participating groups and they are centered around two types of search tasks that are evaluated in INEX: content-only (CO) and content-and-structure (CAS) tasks (see Section 4.1.1). The CAS task had two interpretations in the past: (1) strict CAS (SCAS) where structured constraints are strictly followed, and (2) vague CAS (VCAS) where structured constraints could be relaxed. Each participating group sends a few topics (e.g., 3 CAS and 3 CO topics in 2004), out of which the most appropriate 30 to 40 CO and CAS topics are selected. The 2003 topic set consists of 30 CAS (topic numbers 61 to 90) and 36 CO (topic numbers 91 to 126) topics and the 2004 topic set consists of 34 CAS (topic numbers 127 to 160) and 40 CO (topic numbers 161 to 200) topics.

The topics are specified in XML, and the topic format is a modification of the one used in TREC (see Figure 6.1 in the following chapter). As shown in the example INEX CAS topic 127, given in Figure 5.2, each topic consists of (besides the topic header that contains topic identifier, type, and number) topic title, description, narrative, and a set of keywords. For CO queries the only difference is that the `query_type` of the topic is `CO`, and that the topic `title` element contains only terms and term modifiers.

While topic description, narrative, and a set of keywords are used as guidelines for specifying the query and for explaining the search request, topic title is actually a NEXI (CO or CAS) formulation of the query. The topic title is the formulation of the search request that most retrieval engines use as an input for performing the search task (see [74]).

5.2.3 Relevance assessments

INEX relevance assessments are based on the definition of relevance in TREC: "If you were writing a report on the subject of the topic and would use the information contained in the document in the report, then the document is relevant." Only binary judgments ("relevant" or "not relevant") are allowed in TREC. Furthermore, a document is judged relevant if any piece of it is relevant, regardless of how small the piece is in relation to the rest of the document. However, such definition of relevance is not directly applicable to structured retrieval.

Following the INEX objective, to assess whether XML elements are relevant or not to a given query, two relevance dimensions are introduced. These are: exhaustivity and specificity. Each XML element in the collection can be marginally (1), fairly (2), or highly (3) exhaustive or specific, or not relevant (denoted with pair $(0,0)$), with respect to a given query. For example, in case the user searches for a 'section' element about 'information retrieval', highly exhaustive and highly specific 'section' element is the one that discusses many aspects of information retrieval and nothing beside them. A highly exhaustive and marginally specific 'section' element would be the one discussing many aspects of information retrieval but also many aspects of other topics, e.g., databases. Similarly, a highly specific and marginally exhaustive 'section' element would be the one discussing only some aspects of information retrieval but nothing besides them.

The distinction between the two dimensions in the assessment process is due to the hierarchical organization of XML documents. Unlike in flat text retrieval with binary relevance assessments, if an XML element is considered to be relevant, one cannot say that its parent or child elements are also relevant? Are they more, less, or equally relevant to the original element? That is where two dimensions come into play. A parent, containing usually more (relevant or non-relevant) information than a child, can be more exhaustive but not more specific than a child element. Similarly, a child element can be more specific but not more exhaustive than its parent element. These rules, along with graded assessments, enable relatively simple and intuitive assessment process.

For most metrics, to produce the final evaluation result, e.g., recall-precision graph, the two dimensional relevance assessments are mapped to one dimensional relevance scale by employing a quantization function, $f_{quant}(e,s) : ES \rightarrow [0,1]$. $ES = \{(0,0),(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)\}$ denotes the set of possible assessment pairs (e,s). How this function is incorporated into the evaluation metric is shown below.

5.2.4 Evaluation metrics

Following the TREC [216] paradigm, the effectiveness of information retrieval systems is usually measured by the combination of *precision* and *recall*. Precision is defined by the fraction of the retrieved items that is actually relevant (Equation 5.1). Recall is defined by the fraction of the relevant items that is actually retrieved (Equation 5.2).

$$Precision = \frac{\text{number of relevant elements retrieved}}{\text{number of elements retrieved}} \tag{5.1}$$

$$Recall = \frac{\text{number of relevant elements retrieved}}{\text{total number of relevant elements}} \tag{5.2}$$

Although precision and recall are defined for sets of items, they are in practice used on ranked lists of documents. One approach (that is used in TREC) is to report the precision of documents at several document cut-off points, that is, the precision at 10 documents retrieved, at 20 documents, etc. Furthermore, it makes good sense to average the precision at n documents retrieved on a number of queries, to arrive at an average precision at n documents over, i.e., 50 queries. Averaging over queries is essential, since we cannot possibly draw conclusions on the performance of the system on one query only. This average precision is what we also report in this chapter but using a slightly different metric for the reasons discussed below.

The problem of XML retrieval evaluation

Recent papers [57, 79, 111] have shown that in INEX the issues of element independence and uniqueness are not studied in detail, which as a result introduced problems in the evaluation of search results. Problems are related (1) to the specific organization of the collection, consisting of tree-structured XML documents, and (2) to the richness of user requests – a user can now provide structured hints to the system, including the structured relationship. This is why INEX had difficult time defining the ultimate measure such as precision-recall one for TREC, and why every year different measures arise for evaluating XML retrieval [57, 162, 168, 169].

Among these problems the biggest one for ad-hoc retrieval is the problem of overpopulated recall base and element overlap [111][1]. Overpopulated recall base is the term that is used to denote the fact that in the INEX relevance assessments set a number of nested (overlapped) relevant XML elements exists. Not taking this into account, results in the evaluation metric that rewards retrieval systems

[1]Other problems are not of major importance for the structured retrieval tasks discussed in this thesis. They include the incorporation of following information into the evaluation metric: size of XML elements [79], the time for reading XML elements [57], and user browsing behavior when searching XML collection [169].

that retrieve all the (relevant) elements on the path to ones that retrieve the most relevant elements on the path. To cope with these issues INEX proposed effort-precision gain-recall (ep/gr) [110] measure (for more detailed discussion of INEX metrics see [98]).

Although it does not take into account the overlap, we use *inex_eval* (also called INEX 2002) metric for the evaluation of our document component retrieval experiments for a number of reasons. First of all, it was the official metric for the INEX 2003 and 2004 evaluation, and these are the two test collections that we use in our experiments. Second reason is that we mainly focus on CAS experiments where we treat structured constraints as strict (SCAS). For such experiments the overlap is not a predominant issue as it is up to 26% as reported later in this chapter. Third reason is that *inex_eval* partially incorporates user behavior using estimated search length concept (see below). Finally, by using another metric we would not be able to compare our results with the results of other systems testing their effectiveness on INEX 2003 and 2004 test collections.

Evaluation using inex_eval

The *inex_eval* (INEX 2002) metric computes the so-called *precall* measure, proposed by Raghavan et al. [175], on returned XML elements. It uses the probability that the element viewed by the user is relevant ($P(rel|retr)$):

$$P(rel|retr)(x) = \frac{x \cdot \eta}{x \cdot \eta + esl(x, \eta)} \tag{5.3}$$

In Equation 5.3, $esl(x, \eta)$ denotes the expected search length [42], i.e. the expected number of non-relevant elements retrieved until a recall point x is reached, and η is the total number of relevant elements with respect to a given topic. The expected search length is specified using the formula given in Equation 5.4.

$$esl(x, \eta) = j + \frac{a \cdot i}{c + 1} \tag{5.4}$$

Here j is the total number of non-relevant elements in all levels preceding the final level, a is the number of relevant elements required from the final level to satisfy the recall point, i is the number of non-relevant elements in the final level, and c is the number of relevant elements in the final level. The term level is used here to denote the set of elements that have the same relevance score in the retrieval process, but to be evaluated they are ordered in an arbitrary way. This is termed weak ordering in [42]. Therefore, Equation 5.3 overcomes the weak ordering problem by incorporating arbitrary order of equally ranked elements in the evaluation.

The metric given in Equation 5.3 gives the same values as the precision defined in Equation 5.1 in case the final level has only one (relevant) element. It differs from the former computation of precision in case there are non-relevant elements in the final level that have the same relevance score (rank) as the relevant ones.

Two quantization functions are used for mapping relevance dimensions: f_{strict} (Equation 5.5) and $f_{generalized}$ (Equation 5.6). These mappings are used to determine the number of relevant elements for a query and precision at different recall points. The strict quantization function is used to evaluate retrieval methods with respect to their capability of retrieving highly exhaustive and highly specific XML elements, while generalized quantization rewards methods that retrieve XML elements according to their degree of relevance.

$$f_{strict}(s,e) = \begin{cases} 1 & \text{if } e = 3 \text{ and } s = 3, \\ \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

$$f_{generalized}(s,e) = \begin{cases} 1 & \text{if } (e,s) = (3,3), \\ \\ 0.75 & \text{if } (e,s) \in \{(2,3),(3,\{2,1\})\}, \\ \\ 0.5 & \text{if } (e,s) \in \{(1,3),(2,\{2,1\})\}, \\ \\ 0.25 & \text{if } (e,s) \in \{(1,2),(1,1)\}, \\ \\ 0 & \text{if } (e,s) = (0,0) \end{cases} \tag{5.6}$$

As can be seen in the definition of generalized quantization function, this function favors exhaustivity over specificity. Although other quantizations are possible (see [98]), these two quantization functions are the official ones used within INEX in 2003 and 2004.

To draw the recall-precision graph at every 0.01 recall point, as we do in this chapter, a non-interpolated precall (precision) is computed for every 0.01 recall point using Equation 5.3. Similar to mean average precision (MAP) for TREC (see Section 6.2.1 for more details), we average the precall per each recall level over all queries, and then average it over all recall levels. This *average precision* (*AP*) is what we report in this chapter for most experiments[2].

The feature of *inex_eval* metric is that it calculates recall, based on the assessments set that contains overlapping elements. Additionally, it rewards the retrieval of a relevant component regardless if its part or if it has been seen entirely. This is why we report overlap for some of the runs presented below. We use a set-based overlap defined in INEX 2004 [58] and specified in Equation 5.7. Here, L represents a result list, $overlap(el_1, el_2)$ is true if two elements, el_1 and

[2]We also compared the MAP and AP evaluation results on strict quantization, using *trec_eval* and *inex_eval* respectively, and found no significant difference in the indication of the effectiveness of different retrieval models.

el_2, are overlapping one another, i.e., if they are nested. The overlap is computed for the 1500 elements submitted (by participating groups) for the evaluation.

$$ov = \frac{|\{el_1 \in L | \exists el_2 \in L \wedge el_1 \neq el_2 \wedge overlap(el_1, el_2)\}|}{|L|} \tag{5.7}$$

The *inex_eval* measure produces the evaluation results that can easily be grasped and used for comparison among retrieval systems (similar to TREC recall-precision metric). As strict content-and-structure queries can be considered as a mean to remove overlap (redundancy) from the result list without loosing much precision [98] and based on the reasons already discussed, we employ *inex_eval* for the evaluation of our experimental runs.

5.3 Experimental setup

This section discusses the setup for experimental evaluation of our system on XML retrieval. We start with introducing additional SRA operators used for modeling collection specific search requirements, and continue with the retrieval models used in the experimentation. We end this section with a sketch of the experimental series we discuss in following sections.

5.3.1 Additional SRA operators

To be able to express INEX NEXI queries in SRA we had to extend the operator set discussed in Section 3.2.4, consisting of nine basic SRA operators (depicted in Table 3.2), with two new operators. The first one is a variant of the selection operator (σ). The second one is used for modeling element prior, i.e., the notion that elements do not necessarily have a uniform prior relevance to the query.

To model the selection of regions (elements) that contain numerical content greater, less, equal, greater or equal, and less or equal, than a specified number, we introduce content selection operator $\sigma_{\diamond num}(R_1)$. It is defined in Equation 5.8.

$$\sigma_{\diamond num}(R_1) = \{r_1 | r_1 \in R_1 \wedge \exists r_2 \in C \wedge r_2.t = word \wedge r_2 \prec r_1 \wedge r_2.n \diamond num\}, \\ \text{where } \diamond \in \{=, <, >, \leq, \geq\} \tag{5.8}$$

Such selection is necessary for NEXI queries that include, e.g., selection of articles for a particular year or a range of years (the list of INEX 2003 and INEX 2004 queries in the Appendix A contains a number of such queries). For example, the NEXI topic 65 searches for articles that are published after the year 1998:

```
//article[.//fm//yr > 1998 AND about(., "image retrieval")]
```

Using content selection operator the selection of `yr` elements can be expressed as:

$$\sigma_{>1998}(\sigma_{n=\text{yr},t=\text{node}}(C)) \tag{5.9}$$

To be able to change the default score of elements (1.0), we introduce element prior operator $\nabla(R)$ defined in Equation 5.10. For computing the element prior we used only one specification, i.e., length prior, as given in Equation 5.11. The length prior is based on the assumption that larger elements are more likely to be the right answers to a query.

$$\nabla(R) = \{(r.s, r.e, r.n, r.t, f_{prior}(r)|r \in R\} \tag{5.10}$$

$$f_{prior}^{lp}(r) := r.p \cdot size(r) \tag{5.11}$$

However, other implementations are possible (such as using element level, or statistical information on the distribution of elements and their sizes) and they can easily be incorporated in the SRA element prior operator definition.

5.3.2 Retrieval models

The retrieval models that we tested on effectiveness in this chapter are the ones specified as a composition of score computation, score combination, and score propagation functions, defined in Section 4.2. For score computation we use Boolean model – *Bool* (Equation 4.6), Garden Point XML – GPX (Equation 4.10), language model – LMs (Equation 4.8), Okapi (Equation 4.9), and tf.idf (Equation 4.7). Score combination is implemented as summation – *sum*, product – *prod*, minimum – min, maximum – max, probabilistic sum – *prob* (Equation 4.17), and (only in composition with GPX score computation model) exponential sum – exp (Equation 4.18).

We also use different size computation auxiliary functions, i.e., one that uses region bounds (Equation 4.2) and the other that uses term count (Equation 4.3). Furthermore, we tested whether stemming helps in document component retrieval (see Equation 4.5).

Additionally, as for document component retrieval using the relevance of a "document like" element in generating the final score of a containing element helps in some cases [122, 202], we introduce one more score computation model based on the language model. We call it language model with *article weighting*, and denote it as LMA, as in INEX collection 'article' element plays the role of a document in flat text IR. It is depicted in Equation 5.12. Here α and β are smoothing parameters that regulate the influence of the foreground (element) model (α), document (article) model (β), and background (collection) model ($\mu = 1 - \alpha - \beta$).

$$f_{\sqsupset}^{LMA}(r_1, R_2) = r_1.p \cdot$$

$$\left(\alpha \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{size(r_1)} + \beta \frac{\sum_{r_2 \in R_2 | r \in C \wedge r.n = \text{`}article\text{'} \wedge r_2 \prec r} r_2.p}{size(r)} + \mu \frac{|R_2|}{size(Root)} \right) \tag{5.12}$$

The choice of the ‘document like’ element is not hard coded in the system. It is one of the parameters that can be set in the retrieval model repository (named `context` in Figure 4.3).

Finally, for upwards and downwards score propagation we use the four variants given in Section 4.2.4: average (Equations 4.19 and 4.24), sum (Equations 4.20 and 4.25), and two weighted sum approaches – *wsa* (Equations 4.21 and 4.26) and *wsd* (Equation 4.22 and 4.27). In the experiments we also use smoothed variants of these formulas (Equations 4.23 and 4.28)

5.3.3 Experimental series

The experiments performed on INEX collection are classified into four experimental series. In all series we ’trained’ the models on 2003 collection and test the outcome on 2004 collection. The first set of experiments tests various aspects of relevance score computation: (1) what are the best score computation models, (2) whether stemming helps in the retrieval process, (3) which of the two size computation functions is more effective on document component retrieval, (4) does length prior helps, and (5) what are the best parameter values (Section 5.4).

In Section 5.5 the result of the evaluation of numerous runs with different AND and OR score combination function implementations are discussed. The comparison is done using AP values on different score computation function implementations, comparing recall precision graphs, and comparing the average precision values per query. Finally, score propagation is analyzed in Section 5.6. Different implementations of upwards and downwards score propagation functions are tested on effectiveness. At the end a short discussion of experimental runs is presented.

5.4 Relevance score computation

This section starts with presenting the average precision values obtained on INEX 2003 and 2004 runs with two different element size computation functions in combination with the search where stemming is used or not. Then, the influence of retrieval model parameter values on AP values are analyzed. The experiments are performed with four ‘advanced’ score computation models, i.e., GPX, language models with and without article weighting, and Okapi, as they involve parameters in the specification of score computation (and combination) function. For the size and stemming experiments we use the following retrieval model parameter values: $A = 5$ for GPX, $\lambda = 0.5$ for language model, $\alpha = 0.5$ and $\beta = 0.25$ for language model with article weighting, and $k_1 = 1.5$ and $b = 0.75$ for Okapi.

In this set of experiments we use the fixed choice for element score combination and propagation functions with respect to each score computation model. For the GPX model we use exponential sum for both AND and OR score combination, for language models we use product for AND and sum for OR score combination, and for Okapi we use sum for both AND and OR score combination. Upwards and downwards score propagation functions are in all cases implemented as *sum*.

5.4.1 Stemming and element size computation

The results of the experimental evaluation of advanced retrieval models when stemming is used or not (column *stem*) and when element size (column *size*) is computed using region bounds – *entity* (Equation 4.2) or counting the number of contained terms – *term* (Equation 4.3), are depicted in Table 5.1. The results are reported on INEX 2003 and 2004 collections using strict (column *strict*) and generalized (column *general.*) quantizations, specified in Equations 5.5 and 5.6[3]. The best results for each retrieval model, with respect to element size computation and stemming are given in bold.

By analyzing Table 5.1, we cannot see a clear pattern with respect to the element size computation and the usage of stemming. While stemming seems to have positive impact for language models, it is not clear for GPX and Okapi. Furthermore, for strict quantization and strict assessments (INEX 2003 experiments evaluated using strict quantization) stemming does not seem to work well. This might indicate that stemming is not useful for getting highly relevant elements, but further experiments are needed to confirm this hypothesis.

Looking at different implementations of element size computation function, it seems that counting terms gives better results (see the results on 2003 collection with generalized quantization). In all cases it gives higher AP except for the Okapi score computation model where stemming is not used on INEX 2003 collection with strict quantization. This is confirmed on 2004 collection. However, the differences are in most cases statistically insignificant.

For comparison, the same set of experiments on INEX 2003 and 2004 CO topics, which are actually treated as CAS topics in TIJAH (see Section 4.1.1), does not yield unique conclusions. For some of the models stemming improved the results, and also for some of them counting terms yielded better results. Furthermore, for some of the models, the AP is higher on one topic set and lower on the other.

[3]The AP values for the GPX score computation model are equal for both size computation functions as the GPX score computation formula does not employ the element size computation function (see Equation 4.10).

Table 5.1: Experiments with stemming and computation of element size. We report average precision (AP) values.

			2003		2004	
Model	**stem**	**size**	**strict**	**general.**	**strict**	**general.**
GPX	no	entity	**0.29039**	0.22403	0.07110	0.03240
		term	**0.29039**	0.22403	0.07110	0.03240
	yes	entity	0.28665	**0.22410**	**0.07488**	**0.03481**
		term	0.28665	**0.22410**	**0.07488**	**0.03481**
LMs	no	entity	0.23626	0.21302	0.07126	0.03753
		term	**0.23846**	0.21573	0.07381	0.03871
	yes	entity	0.22786	0.22202	0.07525	0.03805
		term	0.23400	**0.22457**	**0.07787**	**0.03996**
LMA	no	entity	0.24155	0.24370	0.05837	0.03585
		term	0.24800	0.24648	0.05983	0.03669
	yes	entity	0.23902	0.24750	0.06985	0.03959
		term	**0.25370**	**0.25196**	**0.07045**	**0.04015**
Okapi	no	entity	**0.23515**	0.19194	0.05662	0.03250
		term	0.23463	0.19187	0.06121	**0.03380**
	yes	entity	0.20898	0.19842	0.06270	0.03129
		term	0.21107	**0.19873**	**0.06403**	0.03266

By comparing the two versions of language models (with and without article weighting) we can see that while on 2003 runs the language model with article weighting produces better results this is not the case on 2004 runs. Further experiments are needed to analyze this behavior.

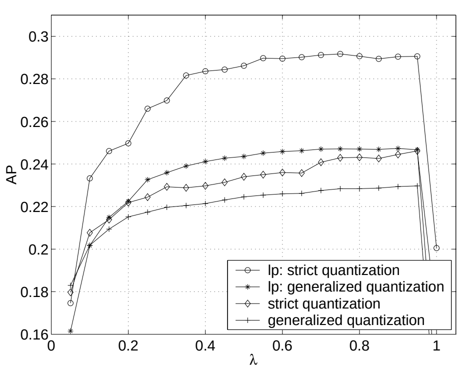
After analyzing the results of using element prior for improving the effectiveness of retrieval models, we found that the only retrieval models for which the effectiveness is improved are language models. Thus we use the element prior in the experiments discussed below. Also, for the rest of our experimental series we use stemming and Equation 4.3 (term count) for element size computation.

5.4.2 Estimating parameters

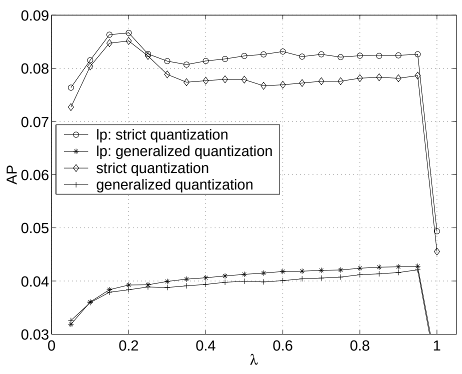
Figures 5.3 to 5.5 depict the AP values for structured retrieval models using different parameter values for each score computation model, except for the GPX model where parameter is varied in the score combination function. The models are first 'trained' on 2003 collection and the outcome is tested on 2004 collection.

Figure 5.3 shows the behavior of the language model based retrieval model, where parameter λ takes the values between 0 and 1. We can see that for the 2003 experiments the AP values tend to get higher the greater the value of λ, using both strict and generalized quantization. This is also the case for language models

Figure 5.3: The influence of different values of λ on AP for the language model:

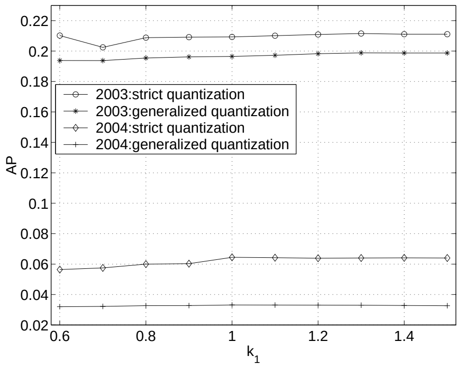


(a) 2003 runs

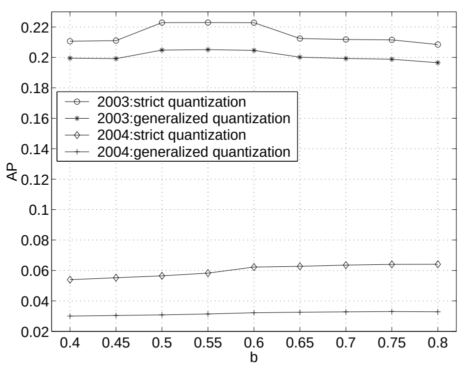


(b) 2004 runs

Figure 5.4: The influence of different parameter values (k_1 and b) on AP for the Okapi model:

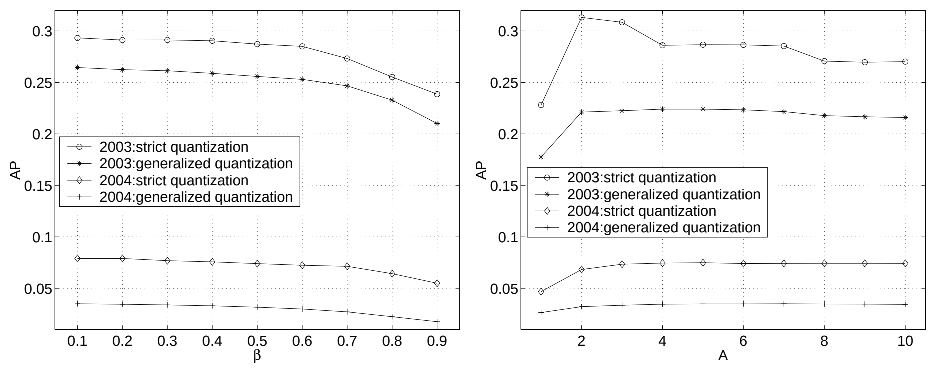


(a) k_1 estimation for $b = 0.75$



(b) b estimation for $k_1 = 1.3$

Figure 5.5: The influence of different parameter values on AP for the LMA (β) and GPX (A) models:



(a) LMA parameter β estimation (b) GPX parameter A estimation

with the usage of length prior (denoted with lp in the figure). We expected the same behavior on 2004 collection.

The expectation was confirmed on 2004 experiments with generalized quantization. However, for the strict quantization, the AP is stable across all values of λ with a small peak around $\lambda = 0.2$. Also a big difference in AP values for 2003 and 2004 experiments can be noticed. This is due to difference in the assessment process. While in 2003 (Figure 5.3a) only the elements that strictly follow the structured constraints are assessed, in 2004 (Figure 5.3b) all elements that are on the relevant element path are assessed (as relevant).

Figure 5.4 shows the AP values for the Okapi based model, where $k_1 = [0.6, 1.5]$, and $b = [0.4, 0.8]$. From 2003 experiments we can see that the AP increases when larger values for k_1 parameter are used, as well as that there is a peak for $b \approx 0.55$. While parameter k_1 shows the expected behavior on 2004 runs, this was not the case for parameter b. On 2004 runs the AP values increase with the larger values of b.

Figure 5.5a shows that the document (article) smoothing parameter should have low values, i.e., $\beta \approx 0.1$. This hypothesis, based on 2003 runs, is confirmed on 2004 runs. For these experiments we fixed the ratio of background and foreground parameters, i.e., $\alpha : \mu = 9 : 1$ ($\mu = 1 - \alpha - \beta$).

The GPX model parameter A gives stable AP values for $A \geq 2$, with the peak around $A = 0.15$ on strict quantization for 2003 runs. This is also confirmed on 2004 runs, but without the peak for strict quantization, as depicted in Figure 5.5b.

Just for comparison we evaluated the CO experiments with the same settings. The best average precision values are obtained in the following cases. For the language model the highest AP values are for $\lambda = 0.4$ to $\lambda = 0.6$, depending on topic set and quantization. The highest AP values for the Okapi model are obtained for $k_1 \approx 0.6$ and $b \approx 0.4$. For the GPX the highest AP is for $A = 2$ on 2003 runs and $A = 1$ on 2004 runs.

Although in some cases we could not confirm that the estimated parameters on 2003 runs give the best results on 2004 collection, we use these parameters for our further experiments. The following values of parameters are used: $A = 0.5$, $\lambda = 0.9$, $\alpha = 0.8$, $\beta = 0.1$, $\mu = 0.1$, $k_1 = 1.3$, and $b = 0.6$.

5.5 AND and OR score combination

In analyzing different instantiations of score combination functions we make a distinction between the models that implement AND and OR score combination using the same function specification, and the ones where these two are different. For the first set of experiments we tried all instantiations of score combination functions. For the second series of experiments we illustrate the best AND score combination functions in composition with different OR score combination function instantiations.

5.5.1 AND and OR instantiated using the same scoring function

Table 5.2 depicts the two best and the worst score combination function in case AND ($\otimes$) and OR ($\oplus$) abstract score combination operators use the same implementation of scoring function. The table shows that the best AND and OR score combination operator implementations (given in bold) follow the specification of the original retrieval models given in Section 2.1 and our hypothesis given in Section 5.1. The best score combination for Boolean, tf.idf, and Okapi models are the ones implemented as sum, while for the language models this is the one implemented as product. Additionally, for the Okapi model and tf.idf model, the implementation of abstract score combination operator as a probabilistic sum also shows nearly the same effectiveness as the one implemented as sum.

From the table we can also see the worst choices for score combination implementations (given in italic). When specifying a Boolean retrieval model one should not use score combination implemented as probabilistic sum. Similarly, for GPX and Okapi score computation models, score combination implemented as minimum should be avoided, while for language models and tf.idf score combination implemented as maximum should be avoided.

The overlap over 1500 returned elements used in the evaluation process is reported on 2004 runs in the last column of Table 5.2 (column *ov*). For all the runs it is below 26%. This indicates that not many overlapping elements exist in the result set; approximately every fourth element returned by the system overlaps with the other returned element.

Table 5.2: Experiments with AND and OR score combination operator where score combination function instantiation is the same for both operators. The two variants of score combination function with the highest AP and the one with the lowest AP are presented, along with the percentage of overlap for 2004 runs.

			2003		2004		
$f_{\sqsupset}$	$\otimes$	$\oplus$	**strict**	**general.**	**strict**	**general.**	*ov*
Bool	prob	prob	*0.04823*	*0.03972*	*0.00841*	*0.00478*	10.73
	prod	prod	0.12526	0.06460	0.01783	0.00858	7.02
	sum	sum	**0.13599**	**0.12064**	**0.04616**	**0.02291**	15.50
GPX	exp	exp	**0.28664**	**0.22410**	**0.07488**	**0.03481**	20.06
	min	min	*0.18742*	*0.11041*	*0.03469*	*0.01799*	14.54
	sum	sum	0.20039	0.14473	0.04320	0.01997	14.54
LMs	max	max	*0.07562*	*0.07398*	*0.01984*	*0.01318*	18.19
	min	min	0.22214	0.16265	0.04772	0.02485	16.90
	prod	prod	**0.26942**	**0.23445**	**0.08224**	**0.04233**	24.50
LMA	max	max	*0.07850*	*0.07460*	*0.01830*	*0.01371*	17.44
	min	min	0.22077	0.19457	0.04843	0.02807	18.24
	prod	prod	**0.27290**	**0.25465**	**0.07970**	**0.04320**	25.30
Okapi	min	min	*0.13369*	*0.08731*	*0.03176*	*0.00815*	6.63
	prob	prob	0.20166	0.18865	0.06186	0.03153	17.38
	sum	sum	**0.22280**	**0.20461**	**0.06222**	**0.03220**	17.27
tf.idf	max	max	*0.13437*	*0.09678*	*0.01888*	*0.00978*	13.32
	prob	prob	0.17780	0.12582	0.04718	0.02135	18.70
	sum	sum	**0.18643**	**0.13160**	**0.04719**	**0.02135**	18.69

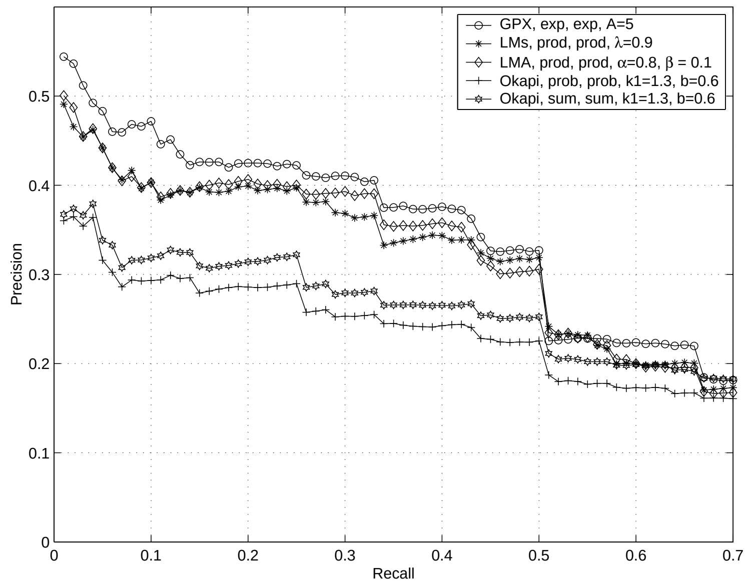
Comparison of recall-precision graphs

In addition to AP comparisons we also compared the *inex_eval* recall-precision graphs for the best score combinations for each model. Two such graphs for strict quantization are depicted in Figure 5.6, for 2003 and 2004 runs. The precision is reported at every (interpolated) 0.01 recall point[4]. The graphs show that GPX based model has a high precision at low recall points. This is especially the case for the 2003 runs with strict quantization and strict assessments (see Figure 5.6a). The precision significantly drops down at the highest recall points, as can be seen for the 2004 runs (and also for the generalized quantization for both runs not presented here).

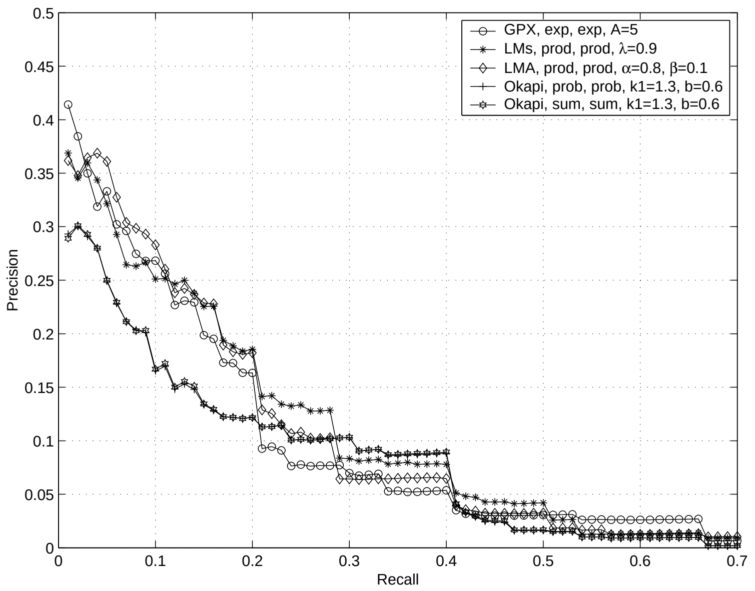
On the other hand, language models and GPX show constantly higher precision than Okapi based models. This can be noticed for 2003 runs up to the recall level of 0.5, and for the 2004 runs up to the recall level of 0.2. Also by comparing

[4]Both graphs are cut at a recall level of 0.7 as the graphs are flat for all models for recall points between 0.7 and 1.

Figure 5.6: Recall-precision graphs for the best AND (and OR) score combination functions on GPX, LMs, LMA, and Okapi score computation model:

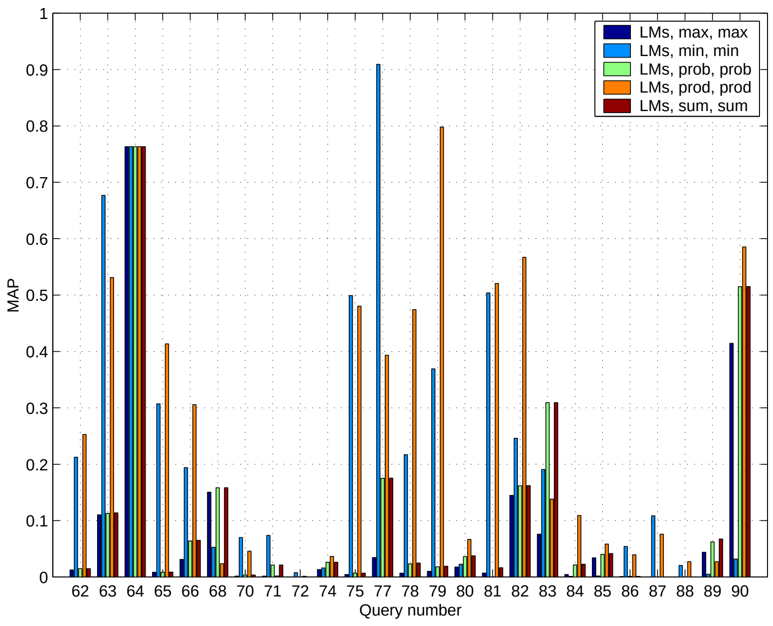


(a) 2003 runs



(b) 2004 runs

Figure 5.7: AP value comparison on language model with different implementations of score combination functions per INEX 2003 query.

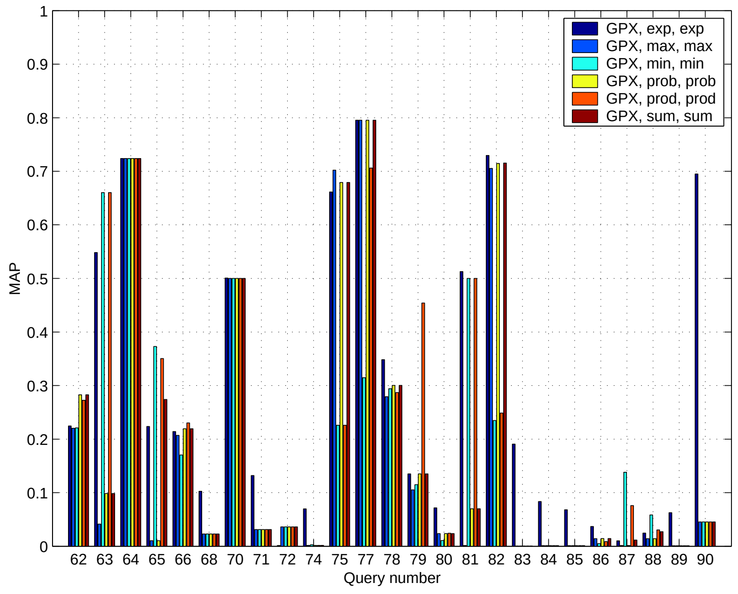


language models with GPX, the precision is slightly worse for language models than the GPX model for low recall points, and nearly the same or even better for the higher recall points. This indicates that the GPX is well suited model for high precision at low recall points for strict scenario (strict quantization, strict assessments), while language models are more suitable for finding all relevant elements (recall oriented) in a more vague scenario.

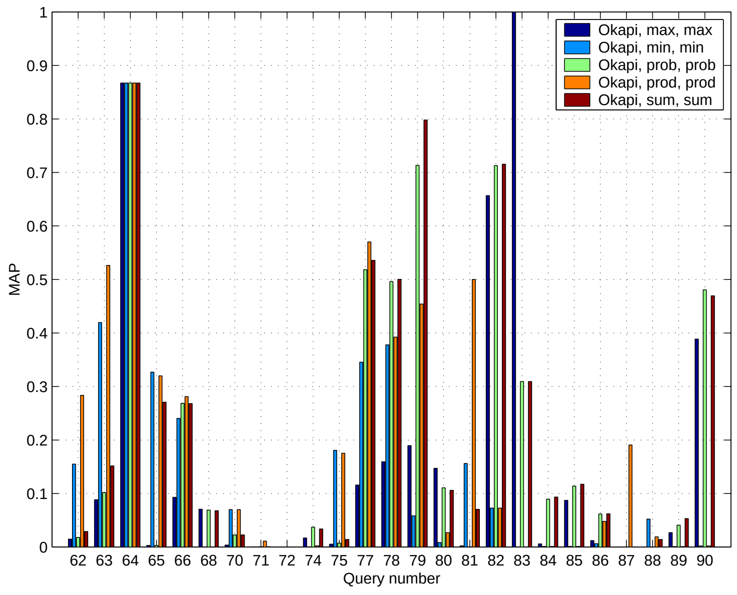
Score combination analysis per query

For the final analysis we compared the impact of different score combination operator implementations on average precision values per query. This is depicted in Figures 5.7 and 5.8. From the first figure we can see that for most queries combination operators implemented as product and minimum, in composition with language models, give higher AP values. Only for few topics other implementations give higher AP values for language models. Among them, for topics that include selection of arbitrary descendant nodes (topics 71 and 72 in INEX 2003 and topic 157 in 2004 topic set), score combination implemented as product is not the best solution. This might indicate that the score combination implemented as minimum is better for this type of queries. However, further experiments are needed to confirm this hypothesis.

Figure 5.8: AP value comparison on GPX and Okapi models with different implementations of score combination functions per INEX 2003 query:



(a) GPX



(b) Okapi

Analyzing the graphs in Figure 5.8 we can hypothesize that the GPX and Okapi based models yield better results if score combination is modeled as product for queries where the answer element is the 'article' (document) element and search is performed either in the 'article' element (GPX), or in the 'article' and elements it contains (Okapi). The first set of topics contains topics 65, 79, 87, and 88. For all of them, score combination implemented as product gives higher AP values than the score combination implemented as exponential sum for the GPX score computation model.

The second set of topics contains topics 62, 63, 65, 70, 75, 79, 81, 82, 87, and 88. For eight of them, Okapi score computation in composition with score combination implemented as product, gives better results than in composition with score computation implemented as sum. Similar behavior can be seen for the topic 137 in the 2004 topic set. Using the paired sign test [193], we can determine that on this set of topics score combination implemented as product in composition with the Okapi score computation model is statistically better than score computation implemented as sum; the probability value of a sign test is 0.0327 which is less than the p value (0.05).

The main conclusion of this effectiveness analysis per query is that different score combination implementations can yield better results for different user requests. Also some retrieval models are more appropriate for some of the queries, and others for other queries. For example, GPX with score computation implemented as exponential sum is usually better for longer queries (queries that contain many terms) than the best performing score combination models for other score computation functions. Although the classification of queries with respect to the implementation of score combination operators cannot be clearly determined, this study indicates that there are relations between the query complexity, type of the user request, and the form of the retrieval model. However, the full analysis of these relations would require much larger topic sets and also more analysis of the user preferences when searching structured collections[5].

5.5.2 Analysis of OR score combination function

The AP values for the two best and the worst instantiation of OR score combination functions for the best (AND) score combination instantiations discussed above are depicted in Table 5.3[6]. The AP values given in bold represent the highest AP for each score computation model, while the ones given in italic represent the lowest AP values. Based on 2003 experiments we could make a hypothesis that the OR score combination for GPX should be implemented as exponential sum, and for language models as minimum. For Boolean, Okapi, and tf.idf score computation models it is not clear which OR score combination implementation gives the highest AP.

[5]That is the task of the INEX Interactive track that started in 2004.

[6]For this set of experiments the overlap does not differ significantly from the previous set of experiments as can be seen in the table.

Table 5.3: Experiments with different instantiations of OR score combination function for the AND/OR score combination function variants with the highest AP in the previous series of experiments. The two OR score combination function with the highest AP values and the one with the lowest AP value are presented.

			2003		**2004**		
$f_{\sqsupset}$	$\otimes$	$\oplus$	**strict**	**general.**	**strict**	**general.**	*ov*
	sum	prob	*0.13230*	*0.11422*	*0.03341*	*0.01670*	14.45
Bool	sum	prod	**0.13875**	0.11944	0.04548	0.02199	14.91
	sum	sum	0.13599	**0.12064**	**0.04616**	**0.02291**	15.50
	exp	exp	**0.28664**	**0.22410**	0.07488	0.03481	20.06
GPX	exp	min	*0.27207*	*0.21607*	*0.06638*	*0.03179*	19.93
	exp	prob	0.28350	0.22042	**0.07538**	**0.03574**	20.02
	prod	min	**0.29344**	**0.25027**	*0.07180*	*0.03771*	23.94
LMs	prod	prod	*0.26942*	*0.23445*	0.08224	0.04233	24.50
	prod	sum	0.29043	0.24733	**0.08243**	**0.04268**	23.91
	prod	min	**0.29648**	**0.27218**	*0.06865*	*0.03859*	24.69
LMA	prod	prod	*0.27290*	*0.25465*	**0.07970**	0.04320	25.30
	prod	sum	0.29301	0.26469	0.07792	**0.04358**	24.81
	sum	min	*0.21144*	*0.19570*	*0.05683*	*0.02918*	16.94
Okapi	sum	prob	0.22217	**0.20557**	0.06189	0.03204	17.28
	sum	sum	**0.22280**	0.20461	**0.06222**	**0.03220**	17.27
	sum	min	*0.18072*	*0.12307*	*0.04121*	*0.01813*	18.67
tf.idf	sum	prob	**0.18643**	**0.13161**	**0.04719**	**0.02135**	18.69
	sum	sum	**0.18643**	0.13160	**0.04719**	**0.02135**	18.69

On the other hand, the worst OR score combination implementations could be clearly identified for each model on 2003 results: probabilistic sum for Boolean, minimum for GPX, Okapi, and tf.idf, and product for language models. However, the 2004 experiments could not confirm many of our hypotheses based on 2003 results. Table 5.3 shows that different compositions of OR, AND, and score computation functions give different results on 2003 and 2004 INEX collections. For example, language models give better results if OR score combination is implemented as minimum on 2003 and as sum or product (which is the worst implementation on 2003 runs) on 2004 collection. Similarly, for GPX based models, the AP values are higher for OR score combination implemented as exponential sum on 2003 collection and as probabilistic sum on 2004 collection.

This also indicates that the OR score combination operator implementation differs with respect to different queries and different tasks. However, for the full analysis we would have to use a collection that contains more queries that have OR statements and not only 9 as in our case (3 in 2003 and 6 in 2004 topic set). Furthermore, user requests have to be analyzed to determine what is the user intention with the OR statement within the structured query.

For our further experiments we use the following implementations for OR score combination functions in composition with different score computation models: sum for Boolean, language models, Okapi, and tf.idf, and exponential sum for GPX.

5.6 Upwards and downwards score propagation

This section illustrates the results of our experiments on different implementations of score propagation functions. The first part compares four implementations of upwards score propagation function (Equations 4.19 to 4.22) and discuss why structured smoothing does not help for component retrieval on INEX collection. The second part shows the outcome of downwards score propagation experiments.

5.6.1 Upwards score propagation

Table 5.4[7] presents the AP values for all score computation models with the best score combination implementations in composition with the four different implementations of upwards score propagation function given in Equations 4.19 to 4.22. As can be seen from the table, our training collection, i.e., 2003 test collection, could not lead to many conclusions, except that for the GPX based model upwards score propagation function should be implemented as sum.

After testing the models on 2004 collection, we can see that for different implementations of score computation and score combination functions different implementations of upwards score propagation give better results. For example, for language models, for each quantization and each test collection, different score propagation implementations give the highest AP.

The only regularity that can be noticed is for the strict quantization AP values for 2004 collection and for the generalized quantization on both collections and across all models (excluding GPX based models). In the former case the AP values are higher for the upwards score propagation implemented using weighted sum normalized by the size of the containing element – *wsa* (Equation 4.21). In the latter case the AP values are higher for the upwards score propagation implemented using average scores of contained elements – *avg* (Equation 4.19) or using a weighted sum normalized by the sizes of the contained elements – *wsd* (Equation 4.22).

Based on these results we can conclude that there might be a pattern in the usage of different upwards score propagation functions for modeling different relevance assessments (vague assessment with strict or generalized quantization). In other words, a relation might exists between the specific user request expressed in

[7]The overlap is in the same bounds as for previous experiments: for Boolean model 15.50% – 16.58%, for GPX 19.98% – 20.19%, for LMs 23.91% – 24.61%, for LMA 24.82% – 25.34%, for Okapi 16.58% – 17.97%, and for tf.idf 18.58% – 18.69%.

Table 5.4: Experiments with different implementations of upwards score propagation function. We report AP values for all four implementations.

				2003		**2004**	
$f_{\sqsupset}$	$\otimes$, $\oplus$	$f_{\blacktriangleright}$	$f_{\blacktriangleleft}$	**strict**	**general.**	**strict**	**general.**
Bool	sum, sum	avg	sum	**0.16811**	**0.14054**	0.05002	0.02397
		sum	sum	0.13599	0.12064	*0.04616*	0.02291
		wsa	sum	*0.12309*	*0.11981*	**0.05898**	*0.02138*
		wsd	sum	0.15986	0.13882	0.05220	**0.02447**
GPX	exp, exp	avg	sum	0.24957	0.21784	*0.07258*	*0.03443*
		sum	sum	**0.28665**	**0.22410**	0.07488	**0.03481**
		wsa	sum	*0.24455*	*0.20945*	**0.09108**	0.03453
		wsd	sum	0.24983	0.21806	0.08095	0.03444
LMs	prod, sum	avg	sum	0.28882	**0.25042**	0.08521	0.04336
		sum	sum	**0.29044**	0.24733	*0.08243*	0.04268
		wsa	sum	*0.27041*	*0.24104*	**0.09116**	*0.04266*
		wsd	sum	0.27931	0.24937	0.08531	**0.04351**
LMA	prod, sum	avg	sum	0.29134	**0.26871**	0.08115	0.04508
		sum	sum	**0.29301**	0.26469	*0.07792*	*0.04358*
		wsa	sum	*0.28032*	*0.25944*	**0.08763**	0.04416
		wsd	sum	0.29233	**0.26871**	0.08127	**0.04511**
Okapi	sum, sum	avg	sum	**0.26550**	0.23573	0.06919	0.03405
		sum	sum	*0.22280*	0.20461	*0.06222*	0.03220
		wsa	sum	0.24326	*0.20414*	**0.07386**	*0.03151*
		wsd	sum	0.25961	**0.23627**	0.06947	**0.03447**
tf.idf	sum, sum	avg	sum	0.20803	**0.14654**	0.05438	**0.02349**
		sum	sum	0.18643	0.13160	*0.04719*	0.02135
		wsa	sum	*0.18120*	*0.12632*	**0.06029**	*0.02082*
		wsd	sum	**0.20848**	0.14594	0.05488	0.02341

the query, e.g., whether the focus is on precise answers or on all possible relevant answers, and the upwards propagation function used.

We also experimented with different values of structure smoothing parameter $\omega = [0, 1]$. However, due to the regular structure of INEX IEEE collection, i.e., since all articles have sections, that have paragraphs, etc., structured smoothing does not help in improving the effectiveness for any of the models. This is also true for downwards score propagation discussed below. In the next chapter we experiment with collections (TREC and CLEF) that do not have a complete structure and we show what is the influence of the structured smoothing parameter in this case.

For testing the downwards score propagation, we selected different implementations of upwards score propagation for different score computation models: average

Table 5.5: Experiments with different implementations of downwards score propagation function. a|s|d stands for *avg*, *sum*, or *wsd*. AP values are reported.

				2003		2004	
Model	$\otimes$, $\oplus$	$f_\blacktriangleright$	$f_\blacktriangleleft$	**strict**	**general.**	**strict**	**general.**
Bool	sum, sum	avg	a|s|d	**0.16811**	**0.14054**	**0.05002**	**0.02397**
			wsa	0.12395	0.11020	0.03876	0.01626
GPX	exp, exp	sum	a|s|d	**0.28665**	**0.22410**	**0.07488**	**0.03481**
			wsa	0.27191	0.21723	0.05930	0.03102
LMs	prod, sum	avg	a|s|d	**0.28882**	**0.25042**	**0.08521**	**0.04336**
			wsa	0.26871	0.23667	0.07368	0.04044
LMA	prod, sum	wsd	a|s|d	**0.29233**	**0.26871**	**0.08127**	**0.04511**
			wsa	0.26580	0.24837	0.07058	0.04052
Okapi	sum, sum	wsd	a|s|d	**0.25961**	**0.23627**	**0.06947**	**0.03447**
			wsa	0.21138	0.19666	0.05596	0.02458
tf.idf	sum, sum	wsd	a|s|d	**0.20848**	**0.14594**	**0.05488**	**0.02341**
			wsa	0.17743	0.12982	0.03603	0.01924

for Boolean and language model, sum for GPX, and *wsd* for language model with article weighting, Okapi, and tf.idf. Although *wsa* gives the highest AP values for most models on 2004 collection with strict quantization, we do not use it as it gives usually the lowest AP values for other evaluations (given in italic in Table 5.4).

5.6.2 Downwards score propagation

Here we present the experimental results on using different implementations of downwards score propagation function. The AP values are given in Table 5.5. Due to the hierarchical structure of the INEX test collection, and the set of topics, three variants of downwards score propagation function implementation, i.e., *sum* (Equation 4.25), *wsd* (Equation 4.27), and *avg* (Equation 4.24), give the same AP values (denoted with a|s|d in the table). The downwards score propagation in most cases propagate the score values from one ancestor element to one descendant element. In this case, *avg* and *sum* just copy the scores of ancestor elements, while *wsd* normalize it by 1 (multiplies and divides by the size of the ancestor element).

However, for the *wsa* instantiation (Equation 4.26), in the propagation process the scores of containing elements are multiplied by their sizes and then divided by the size of a contained element. Therefore, the application of Equation 4.26 for downwards score propagation yields different scores, and consequently different AP values, than other implementations.

As can be seen from the table, on the train as well as on the test collection, in all cases downwards score propagation implemented using Equations 4.24, 4.25, and 4.27 gives higher AP values than downwards score propagation implemented using Equation 4.26. However, other implementations might exist that can further improve the effectiveness of retrieval models. This requires further experiments.

5.7 Discussion

This chapter presents the results of experiments performed to analyze structure information retrieval. The focus is on the component retrieval where an expert user explicitly expresses the structured constrains that should be followed when answering the query, known as strict content-and-structured queries in INEX. The experiments are performed on INEX 2003 and 2004 collections. We tested the effectiveness of different retrieval models instantiated following the four elementary structured retrieval requirements. In Section 5.1 we explained the motivation for the experiments and introduced several hypotheses that we discuss here.

In the first set of experiments we pointed out that the more advanced score computation models, i.e., GPX, language models, and Okapi, give higher average precision (AP) values – higher effectiveness, than the baseline ones, i.e., Boolean and tf.idf. Although we expected that counting the number of terms would yield better results (higher effectiveness) than using region bounds to compute the sizes of elements this could not be confirmed. In many cases, using region bounds for size computations gives better results. The same conclusion can also be made for stemming. While for some models the usage of stemming helps for others it lowers the effectiveness (AP values).

By analyzing the parameter values we noticed that for the Okapi model, the commonly used values for parameters in retrieval are more or less the ones that give high effectiveness for document component retrieval ($k_1 = 1.3$, $b = 0.4$). However, this is not the case for language models. The parameter λ gives better scores if it is larger, i.e., $\lambda \approx 0.9$. For the GPX model, designed for structured retrieval, there is no significant difference for the AP values in the range $2 \leq A \leq 10$.

For the AND score combination function, the best implementations are the expected ones (see Hypothesis 3). The highest effectiveness is achieved when employing AND score computation abstract operator implemented as sum for Boolean, exponential sum for GPX, product for language model, sum for Okapi, and sum for tf.idf score computation model. The comparison of recall-precision graphs for the best composition of score computation and combination implementations pointed out that the GPX model with exponential sum is a precision oriented model, while language models with product are recall oriented. The Okapi model with score combination implemented as sum gave worse results than GPX and language models with score combination implemented as exponential sum or product, respectively.

Furthermore, some compositions of score computation and score combination functions are better on some queries and worse on some other. We illustrated this on the Okapi model where the score combination implemented as product gives better results than the score combination implemented as sum if the answer element in the query is an ‘article’ element and the search is performed in the ‘article’ element or elements inside it. This indicates that experimenting on larger collections might lead to establishing relations between the score combination instantiations and the query structure for producing better results.

However, for the OR score combination, in most cases it is not clear what is the best or the worst OR score combination abstract operator implementation (see Table 5.3). Additionally, some implementations of AND/OR score combinations give the highest AP on 2003 collection, while on 2004 collection they give the lowest.

The score propagation is analyzed in the previous section. It is the most unexplored aspect in structured retrieval. We use several simple formulas for implementing upwards score propagation function. The experiments show that the retrieval model effectiveness varies with respect to the choice of the function. As a result we could not confirm the hypothesis (number 6) that weighted sum produces higher effectiveness than simple sum or average.

Due to the regular XML structure of the INEX collection, for downwards score propagation we can say that it is not useful to normalize the scores of contained regions by their sizes when propagating them from the containing regions. For the more exhaustive analysis, more advanced models have to be defined for score propagation that would include more information from document structure and content (one option is given in Section 7.1 using the element nesting level information).

Finally, we tested how models behave when we use structured smoothing for score propagation. The results show that it is not useful. The document structure in INEX collection strictly follows the DTD, and for the upwards and downwards score propagation contained or containing elements always exist. Therefore, models cannot benefit from the smoothing techniques we implemented. However, other smoothing technique (like the usage of neighboring elements when propagating scores) might improve effectiveness.

In this chapter we illustrated some issues that the researchers should have in mind when implementing structured retrieval models. Also, we show the complexity of the structured retrieval research topic. There are many open questions that need to be answered in the future. The outcomes of presented experiments can be used by other researchers as guidelines when developing their own structured retrieval systems, or when using our algebra/framework for improving the effectiveness of the already existing ones. In the last chapter of this thesis we illustrate some open questions in structured retrieval that need further attention.

Chapter 6

Document Retrieval and Structured Queries

While previous chapter focuses on document component retrieval, the topic of this chapter is improving the effectiveness of document retrieval using query structure and document structure. The chapter starts with the motivation and specifies the hypotheses that we test in the experimental evaluation. The test collections are then presented along with the specification of different structured queries that we evaluate. The chapter continues with the experiments, analyzing score computation and AND score combination on unstructured queries, and OR score combination and upwards score propagation on structured queries. The experimental results are summarized at the end.

This chapter is partially based on a paper published as a Centre for Telematics and Information Technology Technical Report [142].

6.1 Motivation

Although most user queries issued to Web search engines constitute of a limited number of query terms (93% of the Web queries have less than five terms in the query as reported in [103]), these query terms do not have to be the only input that the search engines can use to find the relevant information. For instance, faceted queries [64, 95, 197] utilize knowledge about the semantic relations between words to retrieve more relevant answers, whereas structured queries (discussed in previous chapter) utilize document structure to give more precise answers.

The growth of personalized search, i.e., the information about the user who is performing the search [136, 167], as well as the semantic information that can be deduced from the search request, e.g., using semantic web resources [201], enables easier generation of structured queries. In the future we can expect that the automatic generation of structured queries would be possible, or at least that many retrieval systems will support users in expressing their information need as a structured query. We follow this assumption and focus on analyzing the effectiveness of different retrieval models on document retrieval, using different types of unstructured and structured queries.

We argue that the automatic generation of simple structured queries, such as faceted and field search queries (explained in Section 6.2.2), is more realistic than the generation of long queries such as TREC title + description or title + description + narrative queries. However, it is unclear whether we can achieve higher effectiveness when using these 'simple' structured queries in comparison to the expanded ones (using topic description and narrative).

The question that we try to answer in this chapter is twofold. The first part is about analyzing the behavior of different retrieval models in unstructured and structured document retrieval scenarios. The second part tests whether the formation of structured queries by (end) users, either with or without the help of information automatically extracted from documents, can improve retrieval model effectiveness. In other words, is a search system that is provided with the advanced structured queries (either faceted, field-based, or both) more effective on ad-hoc search task than the same search engine operating on simple or expanded unstructured text queries.

6.1.1 How to compute scores?

Similarly to document component retrieval, the first set of experiments aims at finding out what are the best score computation models for document retrieval. Therefore, the first hypothesis is the same as in the previous chapter.

Hypothesis 1. The effectiveness of more advanced retrieval models, i.e., GPX, language models, and Okapi, should be higher than the effectiveness of simple retrieval models, such as Boolean and tf.idf, for document retrieval.

We also tested the two size estimation formulas for score computation, given in Equation 4.2 and Equation 4.3, and the usage of stemming. Following the experimental results of state-of-the-art retrieval models, we form the following two hypotheses.

Hypothesis 2. For document retrieval, it would be more effective to compute the sizes of elements (documents) by counting the number of terms contained in them then using region bounds.

Hypothesis 3. The usage of stemming should result in higher effectiveness in document retrieval.

Furthermore, we compared the retrieval model parameter estimation for document retrieval with parameters used in state-of-the-art document retrieval models and document component retrieval models (see previous chapter). This set of experiments also tests the usefulness of length prior for the language modeling approach.

6.1.2 How to combine scores?

In this chapter we analyze different implementations of AND and OR score combination functions, in composition with different score computation models. The AND score combination implementations are tested on unstructured queries, while OR score combination implementations are tested on structured (faceted) queries. Besides testing what are the best AND and OR score combination function implementations, similarly as in previous chapter, we make two additional hypotheses.

Hypothesis 4. The effectiveness of faceted (structured) queries should be higher than the effectiveness of short *title* queries as well as the effectiveness of the unstructured version of structured queries (called expanded queries).

To test this hypothesis we experiment with the short few-term queries, expanded queries (short queries plus synonyms), and the faceted version of the expanded queries where query terms are classified into facets and combined in an OR expression. Furthermore, we compare these queries with a long queries formed out of (TREC and CLEF) topic title, description, and narrative. The outcome of the latter comparison tests the following hypothesis.

Hypothesis 5. The effectiveness of faceted (structured) queries should be the same or higher than the effectiveness of long *title + description + narrative* queries.

Proving these two hypotheses on our test collection would show that structuring queries can help in improving the effectiveness of different retrieval models applied to document retrieval.

6.1.3 How to propagate scores?

The final set of experiments tests the usefulness of semantically annotated parts of documents for document retrieval, through different instantiations of upwards score propagation function.

Hypothesis 6. The usage of semantically tagged document components in enhancing document relevance score computation should result in higher effectiveness on document retrieval task.

We compare the effectiveness of different implementations of upwards score propagation on document retrieval to the effectiveness of document component retrieval. Furthermore, due to the incomplete semantic markup of the test collections (see Section 6.2.2) we test whether structured smoothing improves the effectiveness of document retrieval. The structured smoothing parameter ω is estimated on different score propagation functions, and with respect to different score computation models.

The experimental results in analyzing the usefulness of query structure and document structure for document retrieval, as well as in estimating the best implementations of score computation, score combination, and score propagation functions, are discussed at the end. We also point out which of the models are most robust and most effective for document retrieval, with respect to different query types.

6.2 Test collections and experimental setup

This section introduces the test collection and describes the structured query formulation in the first part. Then the retrieval models used in the experimentation are mentioned and the experimental series discussed in sections to follow are explained.

6.2.1 Test collections

For our document retrieval runs we use two test collections: TREC and CLEF. They are discuss below, along with a short description of the TREC and CLEF evaluation initiatives.

TREC

The Text REtrieval Conference (TREC) [216] has started in 1992 with the purpose to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. The goals of the initiative are:

- to support information retrieval research
- to increase communication among industry, academia, and government
- to enable faster integration of research into commercial products
- to increase the availability of appropriate evaluation techniques.

Each year U.S. National Institute of Standards and Technology (NIST) provides a test set of documents and topics (queries). Participants test their retrieval systems on this test set and send the top-ranked results to NIST. NIST pools the individual results, judges if the retrieved documents are relevant or not, and evaluates the results. Finally, each year a workshop is organized that serves as a forum for sharing experiences among research groups.

A set of tracks is established each year that focuses on the existing problems or new research areas. Each track has its own goals but it might share data sets and evaluation strategies with other tracks. TREC started with the Ad-hoc track and for the year 2006 seven tracks are active: Enterprise track, Genomics track, HARD track, Question Answering track, Robust Retrieval track, SPAM track, and Terabyte track. Here we present Ad-hoc track as it is the base for all other tracks and as we use it in our experiments.

The Ad-hoc task has been at the heart of TREC evaluations since the TREC 1 (up to TREC 8). The goal of the Ad-hoc track is to test the effectiveness of different retrieval systems measured in terms of combination of precision and recall (see Equations 5.1 and 5.2 in Section 5.2.4). The data collection consisted of newspaper articles and other documents (500,000 to 700,000 documents) in roughly two gigabytes of text. The participants were given a set of fifty queries posed by real users. Using their retrieval systems participants rank documents from the collection for every query, and the top 1,000 documents for each query are returned to NIST for evaluation. The assessors judge the top 100 to 200 documents from every system for relevance and these results are used for the evaluation.

For the experimental evaluation we use four sub-collections from the TREC 6 collection: Foreign Broadcast Information Service (*fbis*), Federal Register (*fr94*), Financial Times (*ft*), and Los Angeles Times (*latimes*). The topic set consists of TREC 6 topics 301 to 350.

For the evaluation we use the TREC relevance assessments and *trec_eval* evaluation tool to test the effectiveness of distinct query specifications and different retrieval models. The evaluation is based on precision and recall definitions given in Equations 5.1 and 5.2 (in Section 5.2.4). In TREC, precision at several recall points is reported, e.g., precision when the system retrieved 10% of the relevant documents, precision when the system retrieved 20%. Usually a fixed number of recall points is used: 10%, 20%, $\cdots$, 100%. Precision at each natural recall level for a query might also be computed, e.g., if for a particular topic 20 relevant elements exist, the natural recall points would be $\{0.05, 0.10, 0.15, ..., 0.95, 1.0\}$. If these measures are averaged per query, and the resulting measure is averaged over all queries, so-called *mean average precision* (*MAP*) is computed. This is what we report for our experiments in this chapter.

CLEF

The Cross-Language Evaluation Forum (CLEF) [165] also started as a track at TREC (2000-2002). Later it became a separate forum. The goals of the forum are twofold:

- developing an infrastructure for testing, tuning and evaluation of information retrieval systems operating on European languages in both monolingual and cross-language contexts
- creating test-suites of reusable data which can be employed by system developers for benchmarking purposes.

Similar to TREC, CLEF now contains a number of tracks among which the most important one is Cross-Language Speech Retrieval track. The test collection consists of spontaneous conversational English speech on which queries are specified in five languages: Czech, English, French, German, and Spanish (in 2005).

The CLEF test collection we use in our experiments consists of two Dutch newspapers: Algemeen Dagblad (*ad*) and NRC Handelsblad (*nh*). CLEF topics 41 to 90, 91 to 140, and 141 to 200 are evaluated in the experiments. Accordingly, we use the CLEF relevance assessments and *trec_eval* evaluation tool to compute the MAP and test the effectiveness of retrieval models.

6.2.2 Query formulation

To test the effectiveness of different retrieval models on document retrieval we use unstructured (list-of-term) and structured (faceted and field-based) query formulations. Unstructured queries are formed using TREC and CLEF topic title, description, and narrative. Additionally, we use the expanded title queries. We choose three directions for exploring the potential usage of query structure to test whether we can make the retrieval more effective in comparison to the list-of-term queries (ranging from short title queries to long title + description + narrative queries):

- (re)formulating the queries using *faceted* (Boolean) query formulation
- (re)formulating the queries using the document structure and the classification already present in structured documents
- combing the queries that utilize document structure with the simple title and faceted queries.

Query formulation for different query types is explained on the TREC ad-hoc topic 305 depicted in Figure 6.1.

Unstructured queries

In the analysis of unstructured querying we use four types of unstructured queries. These are: title (**T**), title + description (**TD**), title + description + narrative (**TDN**), and expanded queries (**E**). Title, title + description, and title + description + narrative queries can easily be formed using TREC and CLEF topic specifications (see the example TREC ad-hoc topic 305 in Figure 6.1).

On the other hand, expanded queries are usually formed by manually or automatically extending the topic title with more query terms. If we assume that the user is willing to trade his time/effort in stating his query using more query terms for effectiveness, or an automatic way to expand the query exists, e.g., using WordNet [147], routing [24], or relevance feedback [191], a simple title of the TREC ad-hoc query 305 can be transformed into a non-structured query that looks like:

```
vehicles crash cars crashworthy death danger
```

Figure 6.1: An example TREC ad-hoc topic 305 specification.

```
<top>

<num> Number: 305
<title> Most Dangerous Vehicles

<desc> Description:
Which are the most crashworthy, and least crashworthy,
passenger vehicles?

<narr> Narrative:
A relevant document will contain information on the
crashworthiness of a given vehicle or vehicles that
can be used to draw a comparison with other vehicles.
The document will have to describe/compare vehicles,
not drivers.  For instance, it should be expected
that vehicles preferred by 16-25 year-olds would be
involved in more crashes, because that age group is
involved in more crashes.  I would view number of
fatalities per 100 crashes to be more revealing of
a vehicle's crashworthiness than the number of
crashes per 100,000 miles, for example.

<top>
```

In our approach we use manual query expansion (see [64]) based on a faceted query formulation (see below). Section 6.4 explains in more details the effects of manual query expansion, as well as the consequence of using the topic description and narrative in the query formulation, on retrieval effectiveness.

As NEXI query language is used in TIJAH, we transformed unstructured queries (*uq*) into structured ones by enclosing them in the following NEXI expression `//DOC[about(., uq)]`. For example, the expanded query 305 looks like:

```
//DOC[about(.,vehicles crash cars crashworthy death danger)]
```

Faceted query formulation

The origin of this type of Boolean query formulation can be found in the generation of faceted queries [64, 93] that librarians used in early days of information retrieval, even before the computer era. However, not many searchers use and not many retrieval systems support this kind of query formulation. We try to explore whether faceted search should be supported in retrieval systems, i.e., if it is more effective than the usage of list-of-term queries.

Faceted queries are designed based on the following approach. The user first inspects the request and identifies what are the most important keywords in his request. Than he groups these keywords into facets. As an example, for our TREC topic 305 given in Figure 6.1 we can group the following terms:

```
vehicles crash cars crashworthy death danger
```

in two facets:

```
{(vehicles, cars), (crash, crashworthy, death, danger)}.
```

The final step is combining terms from the same facet using the OR operator, and combining these faceted expressions using the AND operator. In such a way we obtain the NEXI query that looks like:

```
//DOC[(about(., vehicles) OR about(., cars)) AND
         (about(., crash) OR about(., crashworthy) OR
                          about(., death) OR about(., danger))]
```

So far not many explicit proofs exist that this kind of query formulation is beneficial for query evaluation, except for some results presented in [95]. However, there are numerous techniques that use the same idea, except that the faceted query formulation is hidden in the layer of query routing and refining, starting with the work of Buckley et al. [24] and Xu and Croft [221]. Additionally, not many models are tested for their robustness with respect to the faceted query formulation. We elaborate more on this issue in Section 6.5.

Thanks to Schiettecatte [197] we were in position to use faceted (Boolean) queries on the TREC collection. We use the whole set of 50 faceted queries for TREC (the complete list of TREC queries can be found in Appendix B). As we did not have such queries for the CLEF collection we developed them ourselves (see the following section) for CLEF 41-90 and CLEF 91-140 topic sets (the complete list of CLEF queries can also be found in Appendix B).

Using document structure for query formulation

Structured IR queries express a combination of searches in different parts of (hierarchically) structured documents (see e.g., Appendix A for XML query examples). In the case of shallow hierarchical documents, such as the ones in TREC and CLEF collections, structured queries can be formed by adding one or more field-like structured constraints to the unstructured query. Looking at the description of the ad-hoc topic 305 in the TREC collection and knowing what are the most frequent terms within SUBJECT elements in the collection, we can come up with the following structured query, where the first part is actually the topic title:

```
most dangerous vehicles
SUBJECT: safety automobile accidents
```

Such queries provide the base for exploring the usefulness of structured information in poorly structured documents, such as TREC and CLEF data collections, and for finding what is the best way of incorporating this additional information with the traditional document retrieval.

To structure the queries we first analyzed the two selected document collections to see what kind of useful information we can extract and how we can use it to help the user in stating his query. In our experiments with field-based queries we only use the LA Times sub-collection of TREC 6 and the complete CLEF data collection as they come with the appropriate structure. The other parts of the TREC collection are not used as they either do not contain meaningful structured elements or it is too difficult to utilize information contained in the structured elements for the query formulation.

The LA Times collection contains three types of elements (tags) that can potentially be used for structured search: SUBJECT, SECTION, and TYPE. However, except for the SUBJECT part, the other two were not suitable for query formulation as the keywords present in these tags are not well classified and they are difficult to utilize by a user when forming a query. Additionally, there are many keywords occurring only several times and a few very frequent ones. On the other hand, the CLEF collection comes with a DTD and is far better organized. Among others, it also contains four informative elements that can be used for making structured queries: GEO (location), HTR (keywords), SEC (section), and PER (persons). Furthermore, unlike in the LA Times collection, tags are populated by a predefined, more thoughtfully chosen, set of terms.

A potential problem for experiments with field-based queries is that in both collections only a part of the documents contains these useful tags. For example, the TREC collection has a SUBJECT tag in 46 849 out of 131 896 documents. It is similar with GEO, HTR, and PER tags in the CLEF collection. Despite this, we decided to use it in our experiments, arguing that well organized collections, with the complete and consistent document annotation, would probably give even better results on structured queries.

For the TREC collection we formed structured queries in two steps. We first select the most frequently occurring (more than 500 times) keywords in the SUBJECT tags. Then, for each query, we choose the terms that are most relevant to the query and add them to the original or faceted query (see Appendix B for the list of faceted queries). For example, the original TREC query 305 with added structured constraints looks like:

```
//DOC[(about(., most dangerous vehicles)
         AND about(.//SUBJECT, safety automobile accidents)]
```

We added structured constraints to 41 out of 50 TREC queries.

To make structured queries out of CLEF topics we asked several of our Dutch colleagues to complete advanced search forms similar to the advanced search forms in, e.g., Google. Our "users" would first read the CLEF topic description and narrative and fill in the following:

- what would be an exhaustive query that they will issue (using TREC faceted queries as an example)?
- what are the persons involved in the query?
- select from a drop-down box whether the topic is about domestic issues ("binenland") or foreign issues ("buitenland")?
- select from a drop-down box in which section of the newspaper the article is likely to be found?
- select from a drop-down box where is the location of the event (continent and country/city)?
- select from a drop-down box what are the keywords that could help the search?

For all the fields with drop-down boxes, an additional "no preference" option is provided. The options for section, locations, and keywords are automatically extracted from the collection. The "searcher's" input is transformed into a faceted query, as stated in the previous section, and a field-based query. For example, for CLEF topic 60 the field-based structured query expressed in NEXI looks like:

```
//DOC[about(., de franse corruptieschandalen) AND
        about(.//HTR, fraude en corruptie) AND
        about(.//SEC, buitenland) AND
        about(.//GEO, europa frankrijk)]
```

The results on structured search are presented in Section 6.6.

6.2.3 Retrieval models and experimental series

We tested the same retrieval models as in the previous chapter. The only difference is that we do not use the score computation implemented as language model with article weighting. Therefore, for score computation we use Boolean model, GPX, language model, Okapi, and tf.idf. Score combination is implemented as summation, product, minimum, maximum, probabilistic sum, and exponential sum. Upwards score propagation is implemented using Equations 4.19 to Equation 4.22. We do not analyze downwards score propagation as there are no queries in the test set that require the usage of downwards score propagation operator. We also use different size computation auxiliary functions given in Equations 4.2 and 4.3, and stemming.

The experimental series are organized as follows. The analysis start with comparing the effectiveness of different score computation models and estimating their best parameter values and size computation. We also explore whether stemming and document length prior help in improving retrieval effectiveness. Then the

AND score combination implementations are analyzed on title, title + description, title + description + narrative, and expanded queries. This is followed by the analysis of experimental results when using structured (faceted) queries with different OR score combination implementations. The final set of experiments tests whether different implementations of structured queries that utilize document markup can improve effectiveness. This set of experiments also includes the estimation of structured smoothing parameter ω.

6.3 Relevance score computation

This section discusses the experimental results using different score computation models, as well as two different element size computation functions and stemming. Additionally, the best retrieval model parameter values are estimated and compared to the parameters of state-of-the-art retrieval models. The experimentation is performed using the advanced retrieval models. The retrieval model parameters are set to the following values in case of size and stemming experiments: $A = 5$ for GPX, $\lambda = 0.5$ for language model, and $k_1 = 1.5$ and $b = 0.75$ for Okapi.

The fixed choice for AND element score combination with respect to each score computation model is used in the experiments: GPX with the exponential sum, language model with the product, Okapi with the sum. We only use the title of the topics for this set of experiments, and report the mean average precision (MAP) values. For the experiments we use *CLEF 141–200* as a 'training' collection and other collections for testing.

6.3.1 How to compute size and whether to use stemming?

From the experiments on the training set, i.e., *CLEF 141–200* collection, depicted in the last column in Table 6.1, we made a hypothesis that the stemming would help in improving the retrieval effectiveness. The experimental results are in favor of this hypothesis, as in all cases on TREC and CLEF collections, the usage of stemming increases the MAP values (see Table 6.1). The increase goes up to more than 20% for *TREC 6* and *CLEF 41–90* runs.

Although we assumed that element size computation based on region indexes (Equation 4.2) would be less effective than term count (Equation 4.3), especially because most of the flat text retrieval systems so far use the latter, this is shown to be a false assumption on *CLEF 141–200*. The results are confirmed on TREC and other CLEF collections (see Table 6.1)[1]. The only run where this is not the case is the Okapi run with stemming. In other cases MAP improvements are between 0.87% and 4.05%. Therefore, we use the element size computation and stemming for the rest of our experimentation series.

[1]This fact is important since precomputations are not used in TIJAH. As a consequence, element size computation is approximately twice as fast as the term count.

Table 6.1: Score computation experiments using GPX, language model, and Okapi, with and without stemming, and with two size computation functions.

Model	**stem**	**size**	**TREC 301–350**	**CLEF 41–90**	**CLEF 91–140**	**CLEF 141–200**
GPX	no	entity	0.1529	0.2254	0.2782	0.3422
		term	0.1529	0.2254	0.2782	0.3422
	yes	entity	**0.1979**	**0.2707**	**0.2880**	**0.3610**
		term	**0.1979**	**0.2707**	**0.2880**	**0.3610**
LMs	no	entity	0.1625	0.2611	0.2903	0.3457
		term	0.1600	0.2585	0.2819	0.3349
	yes	entity	**0.2118**	**0.3207**	**0.3099**	**0.3681**
		term	0.2093	0.3177	0.3009	0.3538
Okapi	no	entity	0.1650	0.2672	0.2912	0.3457
		term	0.1611	0.2648	0.2832	0.3434
	yes	entity	**0.2093**	0.3244	**0.3127**	**0.3683**
		term	0.2075	**0.3252**	0.3048	0.3579

6.3.2 Parameter estimation

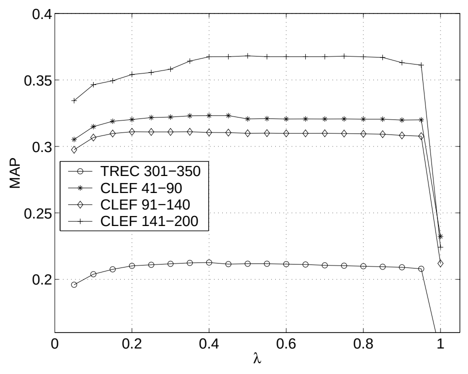
In the following experiments we try to estimate the best values of retrieval model parameters for the language model (λ and the usage of document prior) and Okapi (k_1 and b) score computation implementations, and score combination implementation for the GPX model (A).

For the language model we vary the parameter λ from 0.05 to 1.0, using the granularity of 0.05. For the prior estimation we use the length prior (Equation 5.11). The results are depicted in Figure 6.2. As can be seen, the highest MAP values for the language model are achieved for λ between 0.4 and 0.8, when length prior is not used on CLEF *141–200* run. This is confirmed on other runs, although the MAP values are high from $\lambda = 0.3$ to $\lambda = 0.8$. When using the length prior (see Figure 6.2b) the high MAP is in the range between 0.3 and 0.6 on *CLEF 141–200* experiments, while for other runs we get a similar curve except that the range is wider (0.2 to 0.8).

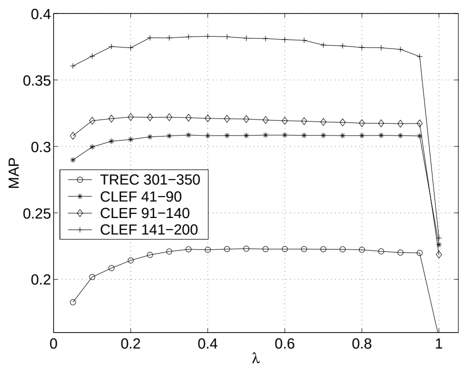
Furthermore, by comparing Figure 6.2a and Figure 6.2b we can see that on three out of four runs the length prior gives higher effectiveness for the best choices of parameter λ. Only on *CLEF 41–90* this is not the case.

In the first set of experiments with the Okapi model we fixed the parameter b to 0.75 and experimented with k_1 set from 0.6 to 1.5 with the 0.1 steps. Figure 6.3a shows that the higher MAP is achieved with lower values of k_1 (0.6 to 0.9) on *CLEF 141–200* runs. The same behavior can be noticed on other runs. By varying b from 0.3 to 0.85 (with a step of 0.05), for $k_1 = 0.7$, as can be seen in Figure 6.3b, the lower values in the range 0.3 to 0.6 give higher MAP values for *CLEF 141–200* runs. This is also the case for *CLEF 91–140* and *TREC 301–305* runs. However,

Figure 6.2: The influence of different values of λ on MAP for the language model:

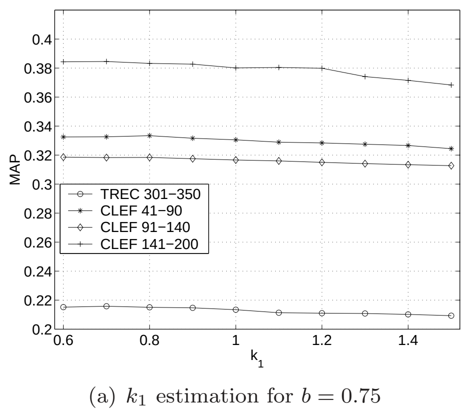


(a) runs without the length prior

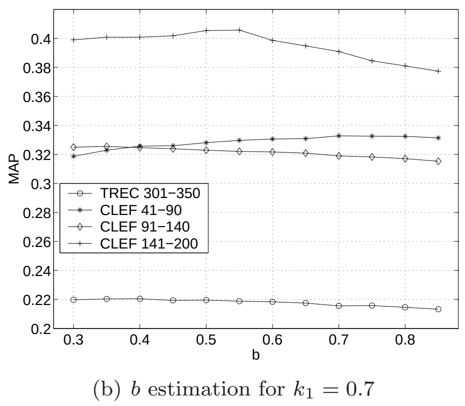


(b) runs with the length prior

Figure 6.3: The influence of different values of parameters k_1 and b on MAP for the Okapi model:

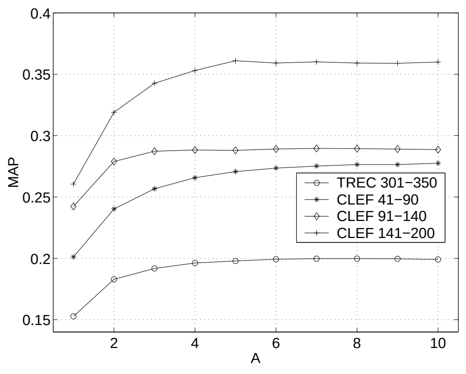


(a) k_1 estimation for $b = 0.75$



(b) b estimation for $k_1 = 0.7$

Figure 6.4: The influence of different values of parameter A on MAP for the GPX model.



for *CLEF 41–90* runs the maximum is around $b = 0.75$. These are relatively low values with respect to the usual values for k_1 (1.2) and b (0.75) in flat text retrieval and also the values for structured retrieval discussed in the previous chapter ($k_1 = 1.3$ and $b = 0.6$). This might be caused by excluding the third factor in the product from the Okapi model (compare Equations 2.8 and 4.9).

For GPX we used only one set of experiments where we employed exponential sum (Equation 4.18) for the score combination operator and estimate the best value of A. As can be seen in Figure 6.4, after recording MAP values for A set from 1 to 10 with step 1, the experiments on *CLEF 141–200* show stable results for $A \geq 5$. This is confirmed on *TREC 301–350*, CLEF *41–90*, and *CLEF 91–140* runs.

In our further experiments we use the following parameters of retrieval models: language model with prior estimation and with $\lambda = 0.4$, Okapi with $k_1 = 0.7$ and $b = 0.4$, and exponential sum for GPX model with $A = 7$.

6.4 Score combination on unstructured queries

The results of our experiments using title (**T**), expanded (**E**), title + description (**TD**), and title + description + narrative (**TDN**) queries are depicted in Table 6.2. In these series of experiments we explore what is the best AND score combination for different score computation implementations and then analyze the impact of distinct query formulations on retrieval effectiveness.

We test five different score combination functions: sum, product, minimum, maximum, and probabilistic sum (Equation 4.17), on Boolean, GPX, language model, Okapi, and tf.idf score computation functions, and exponential sum (Equation 4.18) on GPX. By analyzing the two best performing AND score combination implementations for each score computation function, depicted in Table 6.2, we can

Table 6.2: Unstructured queries: experimental results for the two best-performing AND score combination functions on title (**T**), expanded (**E**), title + description (**TD**), and title + description + narrative (**TDN**) queries.

$f_{\sqsupset}$	$\otimes$	**T**	**E**	**TD**	**TDN**
TREC 301–350					
Bool	min	**0.0803**	0.0317	0.0018	0.0000
	prod	**0.0803**	0.0317	0.0018	0.0000
GPX	exp	**0.1997**	0.1960	0.1308	0.0143
	prod	0.1514	0.0886	0.0734	0.0559
LMs	min	0.1265	0.0467	0.0339	0.0053
	prod	0.2223	0.2205	**0.2230**	0.1878
Okapi	prob	0.2205	0.2232	**0.2239**	0.1587
	sum	0.2205	0.2201	0.2184	0.1551
tfidf	min	0.1004	0.0329	0.0018	0.0000
	prod	**0.1185**	0.0493	0.0019	0.0000
CLEF 41–90					
Bool	min	**0.1288**	0.0681	0.0102	0.0000
	prod	**0.1288**	0.0681	0.0102	0.0000
GPX	exp	0.2752	**0.3137**	0.2535	0.1246
	prod	0.2503	0.1904	0.1776	0.0782
LMs	min	0.1900	0.1025	0.0598	0.0030
	prod	0.3080	**0.3661**	0.3594	0.3302
Okapi	prob	0.3246	**0.3728**	0.3575	0.3240
	sum	0.3257	0.3724	0.3592	0.3215
tfidf	min	0.1541	0.0701	0.0122	0.0000
	prod	**0.1698**	0.0788	0.0133	0.0000
CLEF 91–140					
Bool	min	**0.1223**	0.0051	0.0081	0.0002
	prod	**0.1223**	0.0051	0.0081	0.0002
GPX	exp	**0.2896**	0.2613	0.2813	0.1791
	prod	0.2370	0.0714	0.1127	0.0962
LMs	min	0.2206	0.0374	0.0487	0.0133
	prod	0.3211	0.3370	**0.3866**	0.3658
Okapi	prob	0.3237	0.3499	**0.3944**	0.3923
	sum	0.3247	0.3425	0.3900	0.3889
tfidf	min	0.1600	0.0051	0.0081	0.0002
	prod	**0.1800**	0.0054	0.0095	0.0002

see that advanced retrieval models by far outperform the baseline ones (Boolean and tf.idf). Furthermore, for most advanced models, score combination specified in the original formula of these retrieval models (see Section 2.1) is actually the best AND score combination function. The only exceptions are the tf.idf model where the sum is not in the two best performing combination functions, and the Okapi model where the probabilistic sum implementation (Equation 4.17) shows equally good performance to the score combination implemented as sum (the best runs for all three collections are with the probabilistic sum).

Looking at different query formulations, the effectiveness is much higher for Boolean and tf.idf models on title and expanded queries than on long **TD** and **TDN** queries. This is expected due to the score computation specification for Boolean and tf.idf models (Equations 4.6 and 4.7). However, this is not the case for the advanced models where in most cases the mean average precision values are higher for longer queries (**E**, **TD**, and **TDN**). The MAP values given in bold represent the best run for each score computation model on one topic set.

By comparing original title queries with the expanded ones we can conclude that in some cases the simple title queries outperform the expanded ones (especially for the TREC experiments). In all cases **TD** queries outperform longer **TDN** queries, but it is not clear whether **TD** or title/expanded queries give better results. While on *TREC* topics there is almost no difference in the results (except for the GPX model), on *CLEF 41–90* expanded queries outperform **TD** and **TDN** queries, and on *CLEF 91–140* **TD** and **TDN** queries outperform **T** and **E** queries. This indicates that throwing relevant terms in the query, without properly classifying them, does not necessarily lead to higher effectiveness. The question still remains if the faceted query specification can outperform both title and expanded queries, as well as **TD** and **TDN** queries.

6.5 Score combination on structured queries

The goal of this experimental series is to test how systems can benefit from faceted query formulation and to find the best OR score combination instantiation for the best AND score combination function determined in the previous section. The results for the two best compositions of AND and OR score combination functions for Boolean, GPX, language model, and tf.idf score computation functions, and four in case of the Okapi score computation function, are depicted in Table 6.3. We report four compositions for the Okapi score computation function as it gives high MAP values in combination with the AND score combination implemented as probabilistic sum or sum (see Table 6.2).

If we compare the results of using the expanded queries in Table 6.2 and the results of using the faceted queries in Table 6.3, we can see no consistent improvements in the MAP values for the Boolean, GPX, and tf.idf models, except in the *CLEF 41–90* experiments. However, for the language model and Okapi, for both collections and all three topic sets, there is a consistent improvement over both ti-

Table 6.3: Faceted queries (**F**): experimental results with different OR score combination functions.

$f_\sqsupset$	$\otimes$	$\oplus$	**TREC 301–350**	**CLEF 41–90**	**CLEF 91–140**
Bool	min	max	0.0759	0.1608	0.0665
	min	prob	0.0759	0.1608	0.0665
GPX	exp	exp	0.1849	0.2965	0.2344
	exp	min	0.1603	0.2691	0.2413
LMs	prod	prob	**0.2582**	0.3751	0.3450
	prod	sum	0.2580	**0.3754**	0.3450
Okapi	prob	max	0.2499	0.3857	0.3773
		prob	0.2207	0.3633	0.3217
	sum	max	**0.2559**	**0.3886**	0.3765
		prob	0.2329	0.3711	0.3317
tfidf	prod	prob	0.1166	0.2220	0.0965
	prod	sum	0.1167	0.2220	0.1034

tle and expanded queries. This shows that *structuring queries can help improving effectiveness.*

The paired signed tests show that the improvements are statistically significant ($p < 0.5$) only on TREC collection for: (1) language models with AND score combination implemented as product and OR score combination implemented as sum or probabilistic sum ($p = 0.008$), and (2) Okapi model with OR score combination implemented as maximum and AND score combination implemented as sum or probabilistic sum ($p = 0.016$). This indicates that structuring helps a lot but only for a number of queries. For example, although the improvements for the Okapi model on *CLEF 41–90* collection with AND score combination implemented as sum and OR score combination implemented as maximum improves the effectiveness for more than 10% over **E** queries, the p value is 0.335.

By comparing the MAP values for faceted runs with the MAP values for the **TD** and **TDN** runs on advanced queries we can see that on *TREC* and *CLEF 41–90* topics faceted queries give better results than **TD** and **TDN** queries (given in bold). For example, for the TREC data, the MAP increases up to 16% for Okapi with AND combination implemented as sum and OR combination as maximum, with respect to **TD** and **TDN** (as well as **T** and **E**) queries. However, similarly to the previous analysis, none of the results is statistically significant.

On *CLEF 91–140* the usage of faceted queries does not result in improved effectiveness, but it can be due to the high complexity of manually generated faceted queries. The average length of *CLEF 91–140* faceted queries is 14.81 in contrast to 5.37 in *TREC 301–350* and 7.74 in *CLEF 41–90.* However, *faceted queries show at least equally good effectiveness as long **TD** and **TDN** queries.*

Table 6.3 also shows that more than one OR score combination function exists that has a high MAP on document retrieval: language model with sum and probabilistic sum implementations, and both Okapi models (AND combination implemented as sum and probabilistic sum) with maximum and probabilistic sum as OR combination.

6.6 Score propagation on structured queries

In the last series of experiments we explore how structured information can be incorporated in retrieval models for document retrieval. We test whether we can further improve the precision-recall values by adding field-based structured constraints to the query, using the information that is tagged in a document. The structured constraints are added to the original (*title*) and *faceted* queries.

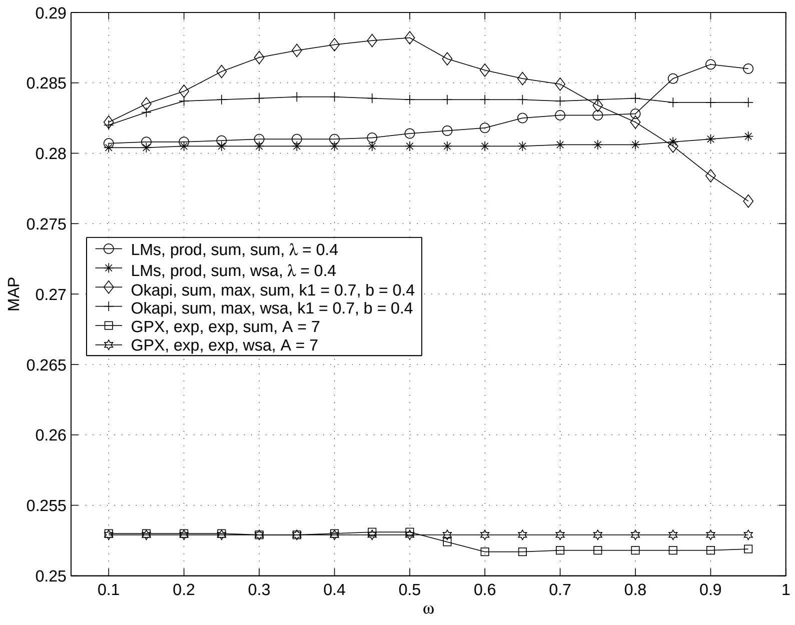
We report three advanced models (GPX, language models, and Okapi) with the best AND and OR score combination functions and using two score propagation functions. The smoothed versions (Equation 4.23 and 4.28) of functions given in Equations 4.20 and 4.21 (*sum* and *wsum*) are employed to test whether and up to what degree we can improve the effectiveness obtained with unstructured queries. Due to the one-to-one score propagation from contained element to the containing element, the smoothed versions of upwards score propagation function implementations given in Equations 4.19 and 4.22 (*avg* and *wsd*), give the same results as the one given in Equation 4.20.

With the first set of experiments we try to find out how structured information should be combined with the “document search”. In other words, what are the values of the structured smoothing parameter ω that give the highest MAP values for different retrieval models. Furthermore we want to find out which of the implementations of the score propagation function are more effective.

We use the *LA Times* collection for estimating ω. The results are depicted in Figure 6.5. Looking at the results, there is no significant difference in using different values of ω. However, for models that use sum-like AND score combination (sum and exponential sum) $0.3 \leq \omega \leq 0.5$ gives slightly better results, while high values of ω are better for the AND score combination implemented as product. Furthermore, in all cases with the high MAP values, score propagation implemented as *sum* gives higher MAP values than score propagation implemented as weighted sum (*wsa*). Therefore, in the following experiments we use Equation 4.20 and $\omega = 0.5$ for GPX and Okapi based models and $\omega = 0.95$ for language model variants.

The mean average precision values of our structured experiments on CLEF collections are reported in Table 6.4. The results clearly indicate that the structured constrains added to the query improve the mean average precision for different variations of language models and Okapi (in 11 out of 12 runs presented in the table) with respect to the title and expanded queries. The best runs (the highest MAP values) are given in bold.

Figure 6.5: The influence of different values of structured smoothing parameter ω on MAP values for upwards score propagation. The *TREC LA Times* data collection is used.



Again, the significance tests shown in the table point out that in most cases the improvements in effectiveness are not significant, except for (given in bold): (1) faceted queries with the language model in comparison to title queries on *CLEF 91–140* collection (2) the Okapi based model on structured + faceted queries in comparison to the Okapi based model on title queries on both collections, (3) the Okapi based model on structured queries in comparison to the Okapi based model on title queries on *CLEF 41–91*, and (4) Okapi based model on structured + faceted queries in comparison to the Okapi based model on faceted queries on *CLEF 91–141*.

Also, by comparing the results from Table 6.4 with the results from Table 6.2, we can see that the effectiveness of structured queries is better on *CLEF 41–90* than the effectiveness of **TD** and **TDN** queries. On *CLEF 91–140* collection the MAP values are almost the same. However, no statistical significance can be noticed if we compare the effectiveness of structured queries with the effectiveness of **TD** and **TDN** queries. This confirms our hypothesis that structuring helps only for particular queries. Therefore, to fully exploit query structuring for improving retrieval effectiveness, detailed analysis on the relation between the query structures (i.e., user requests) and the retrieval effectiveness on a larger test set needs to be performed.

Table 6.4: Comparison of field-based structured queries formed using title (**ST**) and faceted (**SF**) queries, and title (**T**) and faceted (**F**) queries: experimental results on CLEF collections. Beside MAP values we report the outcome of paired sign tests.

$f_{\sqsupset}$	$\otimes$	$\oplus$	$f_{\blacktriangleright}$	**T**	**ST**	**F**	**SF**
			CLEF 41 – 90				
GPX	exp	exp	sum, $\omega = 0.5$	0.2752	0.2872	0.2965	**0.3071**
LMs	prod	sum	sum, $\omega = 0.95$	0.3080	0.3314	0.3754	**0.3853**
paired		**ST** vs. **T**, **F** vs. **ST**, **SF** vs. **F**			*0.240*	*0.101*	*0.664*
sign		**F** vs. **T**, **SF** vs. **ST**				*0.101*	*0.444*
test		**SF** vs. **T**					*0.161*
Okapi	sum	max	sum, $\omega = 0.5$	0.3257	0.3760	0.3886	**0.4313**
paired		**ST** vs. **T**, **F** vs. **ST**, **SF** vs. **F**			***0.016***	*0.336*	*0.101*
sign		**F** vs. **T**, **SF** vs. **ST**				*0.336*	*0.444*
test		**SF** vs. **T**					***0.033***
			CLEF 91 – 140				
GPX	exp	exp	sum, $\omega = 0.5$	0.2896	**0.2965**	0.2344	0.2350
LMs	prod	sum	sum, $\omega = 0.95$	0.3211	0.3161	0.3450	**0.3468**
paired		**ST** vs. **T**, **F** vs. **ST**, **SF** vs. **F**			-	***0.033***	*0.556*
sign		**F** vs. **T**, **SF** vs. **ST**				*0.101*	*0.101*
test		**SF** vs. **T**					*0.101*
Okapi	sum	max	sum, $\omega = 0.5$	0.3247	0.3443	0.3765	**0.3908**
paired		**ST** vs. **T**, **F** vs. **ST**, **SF** vs. **F**			*0.101*	*0.444*	*0.240*
sign		**F** vs. **T**, **SF** vs. **ST**				*0.101*	***0.008***
test		**SF** vs. **T**					***0.033***

The final remark is that the Okapi model with score combination modeled as sum ($\otimes$) and maximum ($\oplus$) is more robust to score propagation and more effective than the language model with all score combination and propagation variants (see Table 6.4 and Figure 6.5).

6.7 Discussion

In this chapter we investigated the composition of different implementations of structured retrieval abstract functions for computation, combination, and propagation of scores. We illustrate how state-of-the-art retrieval models, such as language models and Okapi, can benefit from the formulation of structured queries, having such a flexible structured retrieval framework.

Here we sum up what is the outcome of the experimental runs with respect to the hypotheses introduced in Section 6.1. As expected, the effectiveness of more advanced retrieval models (GPX, language model, and Okapi) is better than the baseline ones (Boolean and tf.idf). Stemming improves the effectiveness in most cases, supporting the hypothesis number 3. Surprisingly, document size computation using start and end document index instead of counting the number of terms in a document showed higher effectiveness.

The parameter values for advanced models were as expected for GPX ($A = 7$) and language model ($\lambda = 0.4$), but for the Okapi model they were slightly lower than usual values for document retrieval ($k_1 = 0.7$, $b = 0.4$). The reason for that might be the employment of slightly different formula then in the original Okapi model (see Section 4.2.2).

We were able to prove that the effectiveness of faceted queries for the best AND and OR score combination implementations on advanced retrieval models is higher than the effectiveness of using title queries with the best AND score combination. However, in many cases long title + description and title + description + narrative queries outperform the faceted ones. Furthermore, the paired signed test shows that most improvements obtained with faceted queries are not statistically significant indicating that faceted formulation helps but only for specific queries.

For structured queries we experimented with different values of structured smoothing parameter. The experiments show that structured smoothing is important on incompletely structured collections, such as TREC and CLEF. The highest MAP values are achieved for structured smoothing parameter in the range $0.3 \leq \omega \leq 0.5$ for models that use sum-like implementations for AND score combination (GPX and Okapi models), and for $w \sim 0.9$ for models that use product in implementing AND score combination (language models).

We also compared unstructured (title and expanded) and structured (faceted and field-based) queries on CLEF test collections. When employing language model and Okapi based retrieval models the effectiveness is improved when using faceted queries, queries that utilize document structure, as well as the combination of faceted and structured queries. Furthermore, the effectiveness of structured queries is equal or better than when using long title + description or title + description + narrative queries. However, most improvements are not significant, pointing out that further research is needed for finding the best way of structuring queries for effective retrieval.

Comparing different implementations of upwards score propagation functions, we illustrate that in structured retrieval summing the scores of containing elements when propagating them to the document (element) gives higher effectiveness then when using weighted sum (similar as for the document component retrieval discussed in the previous chapter).

We have also shown that score computation based on Okapi and language models work well with several score combination functions. For example, Okapi shows good results for AND score combination implemented as probabilistic sum or sum and OR score combination implemented as maximum, while language model shows good results with AND score combination implemented as product and OR score combination implemented as probabilistic sum or sum. Additionally, Okapi based models are more robust to modeling upwards score propagation, i.e., they show greater improvements when using structured queries for document retrieval.

Chapter 7

Flexibility and Extensibility

In previous chapters we discussed how score region algebra is used for modeling textual search, and what is the effectiveness of various retrieval models on structured (document and document component) retrieval. This chapter focuses on score region algebra extensions for using additional document content information in developing new text retrieval models and for modeling retrieval in domains other than text, such as images and videos. We first present the usage of element nesting level information for extending the SRA data model and for developing new retrieval models. How speech transcripts of a video can be utilized by the SRA framework for video retrieval is then discussed. Afterward we focus on image retrieval within the SRA framework. The chapter ends with a short overview of the presented extensions and experimental results.

This chapter is partially based on papers published (1) in the Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX) [144] and (2) in the Proceedings of the 4th Workshop of the INitiative for the Evaluation of XML Retrieval (INEX) [145].

7.1 Using level information

The score region algebra data model is defined in Section 3.2 using five attributes that describe entities in a document structure. However, these five attributes are not the only information that can be used for describing structured documents (see the XML data model discussed in Section 2.3.2). Here we present one example of extending the SRA data model with an additional attribute depicting the nesting level of XML elements. The extension of the data model is explained in the following section, along with the specification of new retrieval models that use this additional level attribute. In Section 7.1.2 we present the experimental results when using new retrieval models on document component retrieval.

7.1.1 Element nesting level in SRA

The score region algebra data model is defined on region sets as depicted in Definitions 1 to 5 in Section 3.2.3. Each region consists of five attributes: region start, region end, region name, region type, and region score. These attributes are sufficient for modeling structured retrieval, following the elementary structured retrieval requirements introduced in Section 3.1, and using adaptations of flat text retrieval models. However, in case we would like to use additional information from structured documents in developing more complex retrieval models, we have to extend the SRA data model.

Here we present the extension of the data model with a *level* attribute. The level attribute denotes the nesting level of a structured entity (element) in the hierarchical document structure. Its introduction requires an extension of Definition 1 in the SRA data model specification. With the additional level attribute, the new definition is as shown below.

Definition 6. *Region tuple r, $r = (r.s, r.e, r.n, r.t, r.l, r.p)$, is defined by these six attributes: region start attribute – s, region end attribute – e , region name attribute – n, region type attribute – t, region level attribute – l, and region score attribute – p.*

Other definitions (Definitions 2 to 5) that specify the SRA data model would remain the same. The domain of region level attribute is the domain of positive integers including 0. The level of 0 is reserved for the collection root node while all other nodes have level attribute value greater than 0, depending how deep they are in a hierarchical document structure.

For the extended data model, definitions of the basic SRA operators in Table 3.2 would remain the same, except that instead of five-attribute regions we would have six-attribute regions (i.e, $(r.s, r.e, r.n, r.t, r.l, r.p)$ region instead of $(r.s, r.e, r.n, r.t, r.p)$ region). However, abstract scoring functions and abstract operators would be able to use the sixth (level) attribute in the specification of score computation, score combination, and score propagation functions. Also new entity selection operators could be defined, e.g., the selection of elements that are at the same level, but this is not the goal of this section. Here we explore the usage of level information for specifying new retrieval models.

As score combination functions most frequently combine scores from regions with the same region bounds, and with the same region level as well, level information is not particularly useful for them. However, it can be exploited in score computation and score propagation functions. The variants for score computation and propagation abstract functions are presented below.

A new score computation function, based on the language modeling approach, that uses the distance between the nesting level of a word's parent element from the level of the containing element, is depicted in Equation 7.1. In the equation max_l denotes the maximal depth (level) of an element in the collection.

$$f_{\sqsupset}^{\mathrm{LMs,level}}(r_1, R_2) = r_1.p \cdot$$

$$\cdot \left(\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot \left(1 + \frac{r_2.l - r_1.l}{max_l + 1}\right)^{-1}}{size(r_1)} + (1-\lambda)\frac{|R_2|}{size(Root)} \right) \quad (7.1)$$

Similarly, we can use the same approach for the other score computation models where the word region score in the score computation instantiation should be multiplied by the factor $\left(1 + \frac{r_2.l - r_1.l}{max_l + 1}\right)^{-1}$. This factor rewards contained elements (words) that are higher in the document structure. For example, if terms are directly inside the containing element, the value of this factor is 1, i.e., $r_2.l - r_1.l = 0$.

We applied the same line of reasoning for the upwards and downwards score propagation. The equations defining the score propagation as sum of scores of descendant/ancestor elements (Equations 4.20 and 4.25) are transformed as depicted in Equations 7.2 and 7.3. This implementation is similar to the one that Sauvagnat and Boughanem introduced in [194].

$$f_{\blacktriangleright}^{\text{sum,level}}(r_1, R_2) = r_1.p \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot \frac{r_2.l - r_1.l}{max_l + 1} \tag{7.2}$$

$$f_{\blacktriangleleft}^{\text{sum,level}}(r_1, R_2) = r_1.p \cdot \sum_{r_2 \in R_2 | r_1 \prec r_2} r_2.p \cdot \frac{r_1.l - r_2.l}{max_l + 1} \tag{7.3}$$

We choose these versions to be tested in our experiments, but equally likely we could implement the level upgrade for other score propagation functions specified in Section 4.2.4. Furthermore, other possibilities exist for incorporating level information in the score manipulation functions, but this would require more extensive analysis that is beyond this thesis.

7.1.2 Experiments with retrieval models that use level info

The usefulness of level information for improving the effectiveness is tested on INEX. We use the same collections as in Chapter 5, i.e., INEX 2003 and INEX 2004 collections. The results of the experimental evaluation are depicted in Table 7.1. The first row ($f_{\sqsupset}^{\text{LMs}}$, $f_{\blacktriangleright}^{\text{sum}}$, $f_{\blacktriangleleft}^{\text{sum}}$) shows the baseline effectiveness (mean average precision – MAP) when employing language model based retrieval model without the usage of level information. For score computation λ is set to 0.5, and we use AND score combination implemented as product, and OR score computation implemented as sum. Upwards and downwards score propagation is implemented as *sum* (Equations 4.20 and 4.25).

The second row shows the results of the same model, except that instead of Equation 4.8 for score computation we use Equation 7.1. The MAP values are slightly lower for the runs that use level information than for the ones that do not use it on both collections and both quantization functions. This indicates that the presented way of incorporating level information in score computation function cannot improve the effectiveness, but it also does not decrease the effectiveness significantly.

Table 7.1: Experiments with language model based retrieval models where score computation and score propagation functions use level information.

Model	2003		2004	
	strict	**general.**	**strict**	**general.**
$f_{\sqsupset}^{\text{LMs}}$, $f_{\blacktriangleright}^{\text{sum}}$, $f_{\blacktriangleleft}^{\text{sum}}$	0.28620	0.24356	0.08233	0.04126
$f_{\sqsupset}^{\text{LMs,level}}$, $f_{\blacktriangleright}^{\text{sum}}$, $f_{\blacktriangleleft}^{\text{sum}}$	0.28498	0.24199	0.08215	0.04121
$f_{\sqsupset}^{\text{LMs}}$, $f_{\blacktriangleright}^{\text{sum,level}}$, $f_{\blacktriangleleft}^{\text{sum,level}}$	0.25566	0.22344	0.07199	0.03800

The usage of level information in score propagation implementations does not lead to higher effectiveness. On the contrary, it significantly worsen the MAP values (as depicted in the last row of Table 7.1), indicating that this way of modeling level information in score propagation functions is not desired. There might be better ways of adding level information to score propagation functions (e.g., see [194]).

7.2 Video search

This section details the extension of SRA for modeling video search. For video retrieval we utilize the automatic speech recognition output of a video. The test collection and speech transcripts are provided by the TRECVID (NIST). The format of speech transcripts is XML, so it fits well in our structured retrieval framework.

The following section describes the test collection and presents retrieval models specially designed for video retrieval, based on the language modeling approach. Then, we compare the effectiveness of these retrieval models with different parameter settings and summarize the results.

7.2.1 Test collection and retrieval models for video search

We first introduce the TRECVID initiative and the test collection we use in our experiments. Afterward, we define the retrieval models.

TRECVID

TRECVID [114] started as a Video track at TREC in 2001 and 2002 and became a separate evaluation initiative in 2003. The goal of the TRECVID is to promote progress in content-based retrieval from digital video via open, metrics-based evaluation. Four tasks are defined in TRECVID: (1) shot boundary determination, (2) low-level feature extraction, (3) high-level feature extraction, and (4) interactive, manual, and automatic search. The data set consists of more than 100 hours of captured TV programs, and topics that specify search for persons, categories of people, specific thing/activity/event, etc. The topics contain, besides textual description of the information need, reference video clips, images, or audio. The evaluation is done based on recall-precision metrics on assessed results.

In our experiments we use the TRECVID 2003 and 2004 collections consisting of CNN World News Tonight and ABC Headline News in MPEG-1 format. We focus only on the automatic search task. For that we use automatic speech recognition output, provided by Gauvain et al. [76]. The output has the form of an XML file assigned to each video.

The collection consists of 121 XML files from the 2003 collection and 128 XML files from the 2004 collection. The data is in MPEG-7 format [127]. The structure of these files is shown in Figure 7.1. A 'Video' element consists of a number of 'VideoSegment' elements, i.e., shots, each depicting the recognized speech from that shot. A shot is selected based on a shot boundary detection algorithm provided by Quénot et al. [173]. Its bounds are depicted in the 'MediaTime' element with its starting time 'MediaTimePoint' and duration 'MediaTimeDuration'. The 'TextAnnotation' element represents the speech transcript record that is enclosed in 'FreeTextAnnotation' tag.

The topic set contains 25 topics in 2003 (topics 100–124) and 24 topics in 2004 (topics 125–148), consisting of few keywords each. The original TRECVID 2003 topic set can be found in [204] and the 2004 topic set can be found in [114]. These topics are transformed into NEXI format where the answer element is the 'VideoSegment' element (see Appendix C for the complete set of TRECVID 2003 and 2004 NEXI queries). For example, the TRECVID 2004 topic 127 that looks like: `Find shots of one or more people and one or more dogs walking together` is transformed into the following NEXI query:

```
//VideoSegment[about(., people dogs walking together)]
```

We performed the evaluation of our video retrieval experimental runs using TREC evaluation metrics, i.e., using the *trec_eval* tool.

Video retrieval models

Video shots are short temporal units, usually consisting of few seconds of video material. The question is whether the usage of spoken information that is contained in these short sequences can lead to high effectiveness. We make a hypothesis that it would probably be more effective to include the information from surrounding shots for the video search. To test this hypothesis we employed several variants of language model based retrieval model (Equation 4.8). We fixed the option for AND score computation and upwards score propagation (due to the query format OR score computation and downwards score propagation operators are not used): AND score computation is implemented as product and upwards score propagation is implemented as *sum* (Equation 4.20).

Besides the language model score computation implementation given in Equation 4.8, we use the two other variants given in Equations 7.4 and 7.5. $r_2 \precsim r_1$ is

Figure 7.1: An example speech transcript from a TRECVID video in XML (MPEG-7) format.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
        xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
   <Description xsi:type="ContentEntityType">
      <MultimediaContent xsi:type="VideoType">
         <Video id="TRECVID2004_100">
            <MediaLocator><MediaUri>19981204_CNNa.mpg</MediaUri></MediaLocator>
            <MediaTime>
               <MediaTimePoint>T00:00:00:0F30000</MediaTimePoint>
               <MediaDuration>PT28M48S18807N30000F</MediaDuration>
            </MediaTime>
            <TemporalDecomposition gap="false" overlap="false">
               <VideoSegment id="shot100_1">
                  <MediaTime>
                     <MediaTimePoint>T00:00:00:0F30000</MediaTimePoint>
                     <MediaDuration>PT2S18078N30000F</MediaDuration>
                  </MediaTime>
                  <TextAnnotation confidence="1.000000">
                     <FreeTextAnnotation></FreeTextAnnotation>
                  </TextAnnotation>
               </VideoSegment>
               <VideoSegment id="shot100_2">
                  <MediaTime>
                     <MediaTimePoint>T00:00:02:18078F30000</MediaTimePoint>
                     <MediaDuration>PT14S6426N30000F</MediaDuration>
                  </MediaTime>
                  <TextAnnotation confidence="0.896090">
                     <FreeTextAnnotation>ALLEGATION AGAINST PRESIDENT CLINTON ON
THE HOUSE JUDICIARY COMMITTEE'S IMPEACHMENT INQUIRY GIVEN ESCAPED DEATH ROW
INMATE BY INDIA'S FREEDOM PRISON EMPLOYEES FOUND OUT THEN WHY A LIGHT TRUCK
BUYERS OF BECOMING THE DRIVING FORCE IN THE AUTO INDUSTRY</FreeTextAnnotation>
                  </TextAnnotation>
               </VideoSegment>
               <VideoSegment id="shot100_3">
                  <MediaTime>
                     <MediaTimePoint>T00:00:16:24504F30000</MediaTimePoint>
                     <MediaDuration>PT6S21201N30000F</MediaDuration>
                  </MediaTime>
                  <TextAnnotation confidence="0.939161">
                      <FreeTextAnnotation>FROM ATLANTA THIS IS C. N. N. HEADLINE
NEWS I'M DAVID GOODNOW THOSE STORIES SHORTLY BUT FIRST THE COUNTDOWN IS
APPROACHING FOR NASA'S</FreeTextAnnotation>
                  </TextAnnotation>
               </VideoSegment>

               ...

            </TemporalDecomposition>
         </Video>
      </MultimediaContent>
   </Description>
</Mpeg7>
```

defined as follows: $r_2 \precsim r_1 \Leftrightarrow (r_1.s - win < r_2.s \leq r_2.e < r_1.s) \vee (r_1.e < r_2.s \leq r_2.e < r_1.e + win)$. It denotes the fact that the region r_2 is in the area around the region r_1 (but not inside it) with the maximum distance between the start or end positions of region r_2 from the end or start positions of r_1 less then win, respectively. λ_1, λ_2, and λ_3 are parameters that control the influence of information found in a shot, surrounding shots (scene), video, and a collection of videos, on the resulting relevance score. $||r_2, r_1||$ denotes the distance between the regions r_2 and r_1 as defined in Equation 7.6.

$$\begin{aligned} f_{\sqsupset}^{\text{LMsFlt}}(r_1, R_2) = r_1.p \cdot (\lambda_1 & \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{size(r_1)} + \lambda_2 \frac{\sum_{r_2 \in R_2 | r_2 \precsim r_1} r_2.p}{2 \cdot win} \\ + \lambda_3 & \frac{\sum_{r_2 \in R_2 | r \in C \wedge r.n = \text{`}Video\text{'} \wedge r_2 \prec r} r_2.p}{size(r)} + (1 - \lambda_1 - \lambda_2 - \lambda_3) \frac{|R_2|}{size(Root)}) \end{aligned} \tag{7.4}$$

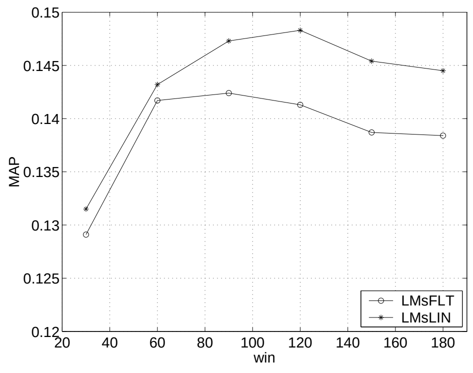
$$\begin{aligned} f_{\sqsupset}^{\text{LMsLin}}(r_1, R_2) = r_1.p \cdot (\lambda_1 & \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{size(r_1)} + \lambda_2 \frac{\sum_{r_2 \in R_2 | r_2 \precsim r_1} r_2.p \cdot ||r_2, r_1||}{2 \cdot win^2} \\ + \lambda_3 & \frac{\sum_{r_2 \in R_2 | r \in C \wedge r.n = \text{`}Video\text{'} \wedge r_2 \prec r} r_2.p}{size(r)} + (1 - \lambda_1 - \lambda_2 - \lambda_3) \frac{|R_2|}{size(Root)}) \end{aligned} \tag{7.5}$$

$$||r_1, r_2)|| = \begin{cases} r_1.s - r_2.e & \text{if } r_2.e < r_1.s \\ r_2.s - r_1.e & \text{if } r_2.s > r_1.e \end{cases} \tag{7.6}$$

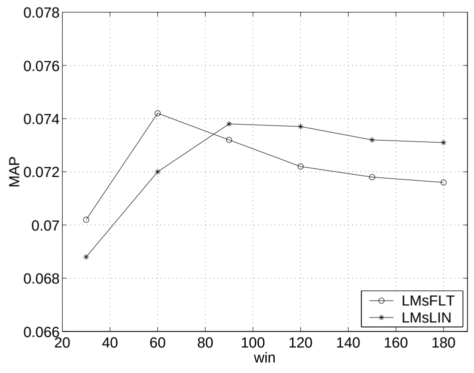
Equations 7.4 and 7.5 use the fact that information from the surrounding shots (context information) can contribute to the improvement of the effectiveness. They follow the reasoning in the "article weighting" approach (see Section 5.3). However, they do not use only the 'document' like evidence, i.e., 'Video' in our case, but the evidence from several surrounding 'VideoSegment' elements, i.e., video shots. The number of surrounding video shot elements that are used in the equations is determined by the size of the window (win). The window actually define what is the estimated size of the video scenes in a video, estimated as a number of tokens (text plus markup). As we were not in position to use video scene segmentation, we experimented with several options for video scene size.

The difference between the two equations is that Equation 7.5 does not take into account the distance of the 'contained' region r_2 from the 'containing' region r_1 when computing scores. However, Equation 7.5 downweights the 'contained' regions that are more distant from the 'containing' region. The linear downweighting is used ($\frac{||r_2, r_1||}{win}$), as depicted in the equation. How effective these models are, also with respect to different parameter settings, is discussed in the following section.

Figure 7.2: Estimating the size of a window (*win*) parameter that defines a video scene based on a number of tokens its speech transcript contains.



(a) 2003 runs



(b) 2004 runs

7.2.2 Experimental evaluation of video search

In the first set of experiments we tested what is the best size for the window (win) defining a video scene. We have chosen the following values to experiment with $win \in \{30, 60, 90, 120, 150, 180\}$. We train the values on the 2003 collection and test them on the 2004 collection. We fixed the values of λ_1, λ_2, and λ_3 to 0.25. The results of this set of experiments are presented in Figure 7.2.

From the 2003 runs we can see that for the 'flat' version (Equation 7.4) of the language model score computation implementation – $LMsFLT$, the best window size is from 60 to 90. As the average size of the shot on 2003 collection is 27.5 this means that we use approximately 3 predecessor and successor video shots for score computation. This is in accordance with the other approaches (see, e.g., [218]). For the 'linear' version (Equation 7.4) – $LMsLIN$, the best window is in the range $90 - 120$. This result is expected as we use linear downweighting for terms in surrounding shots (see Equation 7.5). Similar results are obtained on 2004 collection, given in Figure 7.2b.

In the second experimental set we focus on analyzing the influence of different components in Equations 7.4 and 7.5 on the retrieval model effectiveness. The outcome is depicted in Table 7.2. The results point out that the most influential part of the equations, besides the term statistics inside video shots, is the one that defines the usage of video scene context information (λ_2), given in bold in the table. Furthermore, the 'Video' context information alone (λ_2), given in the last column of the table, is less effective than the usage of background statistics ($1 - \lambda_1 - \lambda_2 - \lambda_3$), given in the second column. This is the case on both, training and test collection.

Table 7.2: The influence of the usage of statistical information from different video components.

Model	λ_1	λ_2	λ_3	**2003 MAP**	**2004 MAP**
$f_{\sqsupset}^{\text{LMsFlt}}$ / $f_{\sqsupset}^{\text{LMsLin}}$	1.0	0.0	0.0	0.0282	0.0188
$f_{\sqsupset}^{\text{LMsFlt}}$ / $f_{\sqsupset}^{\text{LMsLin}}$	0.5	0.0	0.0	0.0657	0.0489
$f_{\sqsupset}^{\text{LMsFlt}}$	0.5	0.5	0.0	**0.1188**	**0.0752**
$f_{\sqsupset}^{\text{LMsLin}}$	0.5	0.5	0.0	**0.1334**	**0.0751**
$f_{\sqsupset}^{\text{LMsFlt}}$ / $f_{\sqsupset}^{\text{LMsLin}}$	0.5	0.0	0.5	0.0640	0.0344

Table 7.3: The influence of parameter values (λ_1, λ_2, λ_3) on retrieval model effectiveness.

			2003		**2004**	
λ_1	λ_2	λ_3	$f_{\sqsupset}^{\text{LMsFlt}}$	$f_{\sqsupset}^{\text{LMsLin}}$	$f_{\sqsupset}^{\text{LMsFlt}}$	$f_{\sqsupset}^{\text{LMsLin}}$
0.10	0.70	0.10	0.1215	0.1321	0.0644	0.0639
0.15	0.45	0.15	0.1381	0.1451	0.0710	0.0717
0.15	0.50	0.10	0.1360	0.1422	0.0706	0.0706
0.15	0.55	0.15	0.1372	0.1428	0.0697	0.0702
0.20	0.40	0.20	**0.1413**	**0.1488**	0.0737	0.0731
0.20	0.45	0.15	0.1387	0.1464	0.0734	0.0732
0.40	0.40	0.05	0.1331	0.1438	0.0761	**0.0757**
0.40	0.45	0.10	0.1384	0.1474	**0.0763**	0.0755

The final set of experiments tests the behavior of these retrieval models when retrieval model parameters are changed. The mean average precision values for these runs are given in Table 7.3. Based on the previous set of experiments we keep λ_3 low (up to 0.2). The best mean average precision values on 2003 runs are obtained for λ_1 around 0.2 and λ_2 around 0.4 (given in bold). On the 2004 runs the same values of λ parameters gave high MAP, but are topped by the runs where λ_1 and λ_2 are around 0.4. However, runs on both collections indicate that for the video models given in Equations 7.4 and 7.5 the emphasis should be on using information from the shot itself and its surrounding shots.

The effectiveness of video variants of the language model is much higher than the effectiveness of the baseline one used for the text search (see the second row in Table 7.2). By comparing the two columns ($f_{\sqsupset}^{\text{LMsFlt}}$ and $f_{\sqsupset}^{\text{LMsLin}}$) from Table 7.3, no significant difference in the effectiveness between the two introduced video shot score computation models can be noticed, although on 2003 runs linear version gives higher MAP values. However, the choice of parameter values plays an important role.

These experiments show that SRA can support video search using speech transcripts. However, speech transcripts are not the only way of modeling video material. Video can be described in terms of video features, that can be used for detecting the video content. Adding feature extractors and using their output for retrieval could further improve the video retrieval effectiveness (see e.g., [218]). As SRA supports content description independence, these features can be modeled at the physical level and used in SRA for new retrieval operators and retrieval models without significant modification of the score region algebra. An example of such an SRA extension is explained in the next section.

7.3 Image search

This section details our approach for image by example search, and explains how this type of search can be integrated with the text search within the SRA framework. In the first part we present the test collection we use in our experiments. Then, the SRA extensions for image search are presented along with the specification of image retrieval models. The section is closed with a short discussion of experimental results.

7.3.1 Test collection and retrieval models for image search

For the experimental evaluation of image search we use the test collection provided within the INEX Multimedia track. The details of the track, the collection data, and topics are given below. Afterward we discuss how we support image search in TIJAH, and how image search is modeled within SRA operators.

INEX Multimedia track

The task of the Multimedia track at INEX is to retrieve relevant document components based on a structured information need expressed over multimedia documents. A structured retrieval approach developed by participants should combine the relevance of different media types into a meaningful ranking. The multimedia track focuses on using the structure of the document to extract, relate, and combine the relevance of different multimedia fragments.

The initiative started in 2005 with the emphasis on combining text and image search. In our experiments we use the test collection from this year. The data collection is based on a small Lonely Planet document set, consisting of 462 XML files. Each XML file contains touristic information about towns, interesting cites, countries, etc. An example of such file is given in Figure 7.3a. Many files also contain some pictures illustrating the region discussed in the file. In the text file this is modeled through the 'image' element, where 'image' attribute 'filename' keeps the relative location of the image (as depicted in the figure). The images are stored on these locations (see Figure 7.3b for an illustration).

Figure 7.3: The INEX Multimedia collection provided by Lonely Planet:

```
<?xml version="1.0" encoding="UTF-8"?>
<destination node_id="627" cobj_id="14644" object_type_id="8" language="eng"
      date_exported="2005-02-23">
   <name>Amsterdam</name>
   <general>
      <full_name>Amsterdam</full_name>
      <introduction>
         <mini>
            <p>History, art, a head of beer and a roll-your-own.</p>
         </mini>
         <short>
            <p>Amsterdam is one of the world's best hangouts, a canny blend of
old and new: radical squatter art installations hang off 17th-century eaves;
BMWs give way to bicycles; and triple-strength monk-made beer is drunk in
gleaming, minimalist cafes.</p>
         </short>
         <medium>
            <p>The city seems to thrive on its funky mix and, despite hordes of
tourists, still manages to feel quintessentially Dutch. The old crooked houses,
the cobbled streets, the tree-lined canals and the generous parks all contribute
to the atmosphere.</p>
         </medium>

         ...

   <images>
      <image filename="/images/BN13629_109.jpg" object_type_id="13">
      </image>
      <image filename="/images/BN1830_4.jpg" object_type_id="13">
      </image>
      <image filename="/images/BN13628_30.jpg" object_type_id="63">
      </image>
      <image filename="/images/BN13629_20.jpg" object_type_id="72">
      </image>

      ...

   <copyright>Copyright Â© 2005 Lonely Planet Publications</copyright>
</destination>
```

(a) an example Lonely Planet file about Amsterdam

BN13629_109.jpg

BN1830_4.jpg

BN13628_30.jpg

(b) example images from Amsterdam

Similar to the INEX Ad hoc track, participants are supposed to develop topics and perform assessments. The topic format used is similar to the NEXI CAS topic format (see Section 5.2). In 2005 participants developed 25 topics (the complete list of INEX Multimedia NEXI queries is given in Appendix D). The assessments process is done similar to the assessments in the INEX Ad hoc track, except that only elements that strictly follow structured constraints are assessed and that binary assessments are used, i.e., XML element is either relevant or not.

The evaluation is done using TREC evaluation tool. Therefore, different approaches are compared using mean average precision and drawing recall-precision graphs. In this section we report only mean average precision values.

Modeling image search in TIJAH

Image search is handled in the same framework as text search, and our three-level database system TIJAH is extended to support it. At the end-user level the NEXI query language is extended for query by example image search, and at logical level new operators are introduced. However, due to different nature of the domain data, images are stored and handled in a different manner than textual XML data at the physical level.

The original NEXI syntax (given in [209]) is extended with an extra token '`src:`' (as shown in Chapter 1) that defines the location of the sample image with which the destination image should be matched. For example, the multimedia query 11 looks like:

```
//destination[about(.//image, fruit vegetables
                              src:/images/BN2787_4.jpg)]
   //point_of_interest[about(., food fruit vegetable market)]
```

The first *about* contains a request for image similarity search. The destination image that need to be matched is `images/BN2787_4.jpg`. In the NEXI processing at the end-user level, the '`src:`' part of the *about* is transformed into *about_image* and its relative path given in the NEXI '`src:`' specification is resolved into the path to the location where data for the image matching is stored. The *about_image* command is then forwarded to the logical level.

To express image search in SRA we extend the SRA operator set with two additional operators modeling image similarity selection ($\sigma^{i \approx sample}_{n=name}(R_1)$) and element-image score computation ($\sqsupset^{i}_{p}$). The $\sigma^{i \approx sample}_{n=name}(R_1)$ operator has a similar definition to the basic score region algebra selection operator $\sigma_{n=name,t=type}(R)$ (see Table 3.2), except that it selects attribute regions and that the score of a region (p) is now computed by a call to an external function $f^i(r_2.n, sample)$. The function $f^i(r_2.n, sample)$ uses information extracted from the sample image and the image that should be selected and it computes the score of an image region based on similarity between the sample image and the selected image. The specification of the image selection operator is given in Equation 7.7.

$$\sigma^{i \approx sample}_{n=name}(R_1) := \{(r_1.s, r_1.e, r_1.n, r_1.t, f^i(r_2.n, sample)) \mid r_1 \in R_1 \ \wedge$$
$$\exists r_2 \in C \ \wedge \ r_2 \prec r_1 \ \wedge \ r_2.t = attr\_val \ \wedge \ r_1.t = attr \ \wedge \ r_1.n = name\} \quad (7.7)$$

Here, *sample* is the location of the sample image data specified with the '`src:`' statement in the NEXI query, resolved in the end-user processing step, C is a set of all regions in the database, *attr* is the attribute node, and *attr_val* is the value of the attribute *attr*.

The operator $\sqsupset^i_p$ is defined in the same way as $\sqsupset_p$ operator (see Table 3.2), except that it allows computing the score of a region that contains images with the usage of different scoring formula ($f^i_{\sqsupset}(r_1, R_2)$) than for terms (e.g., given in Equation 4.8). Its definition is given in Equation 7.8.

$$R_1 \sqsupset^i_p R_2 = \{(r_1.s, r_1.e, r_1.n, r_1.t, f^i_{\sqsupset}(r_1, R_2)) | r_1 \in R_1 \wedge r_1.t = node\} \quad (7.8)$$

Having introduced two new operators in score region algebra we can now express the image search. For example, the *about_image* in the multimedia query 11 (`about(.//image, src:/images/BN2787_4.jpg)]`) is transformed into the next SRA expression:

$$\sigma_{n=\text{image}, t=\text{node}}(C) \sqsupset^i_p \sigma^{i \approx \text{BN2787\_4.jpg}}_{n=\text{file\_name}}(C)$$

As shown in Equations 7.7 and 7.8, we follow the transparency paradigm, i.e., we support content description independence and retrieval model independence. This is achieved through the usage of two abstract functions in the specification of these operators: $f^i(r_2.n, sample)$ and $f^i_{\sqsupset}(r_1, R_2)$. The first one hides how images are modeled and matched, and the second one abstracts away from the way how single image scores are combined in case one element in a structured document contains multiple images.

Image retrieval model

For our implementation we selected an approach of modeling image search based on a generative probabilistic model. At indexing time, we estimated a generative probabilistic model of each of the images in the collection. The feature values (model parameters) are stored in separate tables at the physical level of TIJAH, i.e., in MonetDB. In addition, we constructed a table that links the image identifiers (*sample*) to the corresponding nodes in the collection tree. The image selection operator is implemented as a new MIL function that computes the similarity between each collection image model and the sample image, as defined below.

Similarity between sample images and collection images is estimated using Gaussian mixture models (GMM). Each of the images in the collections ($\mathcal{I}(n_i)$)

is modeled as mixtures of Gaussians with a fixed number of components K. The probability of generating a feature vector $\boldsymbol{x}$ from an image $\mathcal{I}(n_i)$ is defined in Equation 7.9.

$$P(\boldsymbol{x}|\mathcal{I}(n_i)) = \sum_{k=1}^{N_k} P(K_{i,k})\ \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_{i,k}, \boldsymbol{\Sigma}_{i,k}), \tag{7.9}$$

Here N_k is the number of components in the mixture model, $K_{i,k}$ is component k of a class model $\mathcal{I}(n_i)$, and $\mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The feature space of the vectors $\boldsymbol{x}$ is based on the discrete cosine transform (DCT) coefficients obtained from 8x8 pixel blocks. However, as the image retrieval model is not the primary topic in this section, we do not discuss it further. For details of the feature vectors and the GMM approach, see [217].

The score of an arbitrary image in a collection, given a sample image from a query, is determined by the likelihood that the corresponding model generates the feature vectors ($\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$) representing the sample image. As in the language model approach for the relevance score computation (Equation 4.8), we interpolate image 'foreground model' with a 'background model' based on collection statistics, as depicted in Equation 7.10.

$$f^i(n_i, sample) = \prod_{\boldsymbol{x} \in \mathcal{X}_{sample}} (\lambda \cdot P(\boldsymbol{x}|\mathcal{I}(n_i)) + (1-\lambda) \cdot P(\boldsymbol{x})) \tag{7.10}$$

Equation 7.10 defines the abstract function $f^i(n_i, sample)$ used for similarity computation in the image selection operator $\sigma^{i \approx sample}_{n=name}(R_1)$.

In the implementation of the image score computation abstract function for the $\sqsupset^i_p$ operator, we use the multiplication of a left operand region score and the average of contained right operand region scores. The specification is depicted in Equation 7.11. As in the Lonely Planet collection all 'image' elements contain one image, the score computation for the image containment operator results in a score computed as a product of a left operand region score and a right operand region score, i.e., $r_1.p \cdot r_2.p$. For other collections or other tasks different implementations of image score computations might be more appropriate.

$$f^i_{\sqsupset}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{|\{r_2 \in R_2 | r_2 \prec r_1\}|} \tag{7.11}$$

7.3.2 Experimental evaluation of image search

The aim of this set of experiments is to test if using visual similarity can improve the effectiveness of textual search. Furthermore, the experiments are presented also to demonstrate the flexibility and extensibility of our framework with respect

Table 7.4: MAP values for the INEX Multimedia experiments.

Model	**text only**	**text + image**
GPX	0.2567	0.2627
LMs	0.2751	0.2600
Okapi	0.2110	0.2133

to composition of retrieval models in different domains. The MAP values of experimental runs are presented in Table 7.4.

In the experiments we compared the multimedia queries containing only text search (*text only* column) and the ones containing text and image by example search (*text + image* column). The first set of queries is generated by removing (`src:`) clauses from the original INEX Multimedia NEXI queries. For example, the query 11 without the image search looks like:

```
//destination[about(.//image, fruit vegetables)]
   //point_of_interest[about(., food fruit vegetable market)]
```

For the evaluation of these two approaches we use advanced text search models, i.e., GPX, language models, and Okapi. The parameters used are as follows: $\lambda = 0.5$, $k_1 = 1.5$, $b = 0.75$, and $A = 5$. For AND score combination we use exponential sum for GPX, product for language model, and sum for Okapi. OR score combination is implemented as exponential sum for GPX, and as sum for language model and Okapi score computation model. Upwards and downwards score propagation is implemented as *sum* (Equations 4.20 and 4.25).

As can be seen in Table 7.4, while for the language model (the best text search run) image search produce slightly lower MAP values, it is the opposite with the other two text search models. There is no significant difference between the retrieval effectiveness of all three models when using or not using image search. We believe this could be due to the relatively simple image retrieval model we use and due to the simple collection used. Some more advanced image search model on more complex collection might give different results. More research is needed to investigate if and how visual information can help to improve retrieval effectiveness in this and other collections.

7.4 Summary

Throughout this chapter we showed that the score region algebra (and TIJAH database system) is an extensible and flexible framework. It is extensible with respect to:

- modeling additional information describing document content and structure, as well as with respect to modeling different document content types;

- incorporating advanced search techniques, such as new retrieval models and new operators specifying retrieval in domains other than text.

These SRA features are a consequence of SRA supporting retrieval model independence and content description independence.

The flexibility and extensibility of SRA is illustrated on the following three tasks:

1. incorporating element nesting level within the SRA data model and using it to develop new retrieval models;
2. modeling video retrieval by means of developing new text retrieval models for search on structured speech transcripts of video material;
3. search on heterogeneous data sources, i.e., a combination of image and text search

For the first task, SRA data model is extended with the additional attribute depicting the nesting level of elements in a hierarchical document structure. This additional information is used for specifying new retrieval models. In this chapter we present the language model score computation extensions that use level information. Furthermore, we extend the score propagation functions given in Equations 4.20 and 4.25 to use level information. The extension in both cases downweights the scores of containing or contained elements with respect to their relative nesting level distance to the contained or containing elements, respectively.

While the score computation extension resulted in the similar effectiveness as the original model (not using the level information), the propagation extensions had negative influence on MAP values. However, these approaches present only one way how element level information could be used. More advanced models that can improve the retrieval model effectiveness might exist but to discover them further research is needed.

Modeling video retrieval within SRA did not require any extensions of the data model or operator set, as video content is described in terms of XML documents depicting the speech transcripts of videos. However, retrieval models for structured text search presented in Section 4.2, such as the one based on the language modeling approach, resulted in poor effectiveness. To improve the effectiveness we introduced variants of abstract language model score computation function. In these variants relevance score is computed using not only foreground and background statistics, but also statistics from a complete video sequence and from the surrounding shots modeling a video scene. The usage of these additional statistics doubled the effectiveness.

For modeling image (query by example) retrieval, and its fusion with text retrieval, we extended the SRA operator set and added new abstract functions. New operators in SRA specify image selection and element-image score computation. The resulting score attribute in these two new operators is defined throughout two new abstract scoring functions. One defines the assignment of a relevance score to an image, based on its similarity to a sample image. The other defines how score of an element containing multiple images is computed.

The experimental evaluation showed that using a simple image search model, based on Gaussian Mixture Models, in combination with text search, shows no or insignificant improvements when compared to text search models. However, similar to the video retrieval scenario, more advanced image retrieval models or better combination of text and image search models might improve the effectiveness.

Chapter 8

Conclusions and Future Work

With the rapid growth of documents generated using structured format (XML, SGML, HTML, etc.) a new information retrieval research area has been created that addresses the problem of ranked retrieval of elements (document components). Within the area, the goal of structured retrieval systems is not to merely return the most relevant documents based on their content. Their aim is to exploit document structure and semantics of structured annotation of documents to retrieve the most relevant document components.

We focus on the system-oriented aspects of structured retrieval. We try to give an answer to the question what future structured retrieval systems need to support to be *effective* on structured retrieval, as well as to be *flexible* with respect to different retrieval models and different types of content that can be a part of structured documents. To give an answer to this question we performed an analysis of structured queries. We identified four elementary requirements that have to be supported by the structured retrieval framework (see Chapter 3). Guided by these requirements we explain the development of a flexible and extensible structured IR framework, with a logical algebra as its central part.

The algebra (framework) supports transparent instantiation of retrieval models by supporting *retrieval model independence* and *content description independence*. Retrieval model independence denotes the concept that different retrieval models can be specified within the algebra without affecting the end-user applications and the logical algebra itself. Content description independence denotes the concept that different types of document content information can be used for specifying retrieval models, without knowing how this information is stored and accessed at the physical level. The flexible structured IR framework, with the algebra that supports retrieval model independence and content description independence as its central part, is used in this thesis as an experimental platform for developing effective structured retrieval systems applicable to different domains, such as text documents, images, and videos.

The main questions in designing and developing the flexible and extensible structured retrieval framework that motivates the work presented in this thesis are the following three:

- to what extent can a logical algebra that models structured IR support retrieval model independence? (Chapters 3 and 4)

- what is the influence of different retrieval model instantiations within the algebra on the effectiveness of document component retrieval and document retrieval? (Chapters 5 and 6)
- can the algebra be extended to support ranked retrieval using richer data models and to provide effective retrieval in domains other than text? (Chapter 7)

These questions are addressed throughout the development of the *score region algebra* (*SRA*). The algebra is specified in Chapter 3 based on existing region algebra approaches that do not include relevance ranking (Section 2.5). How SRA supports retrieval model independence and content description independence, and how its features can be used to improve the effectiveness is explained in Chapters 4 to 7. Here we only summarize the findings regarding the three research questions in Sections 8.1, 8.2, and 8.3, respectively. For each question we emphasize what is our contribution and what are the potential directions for future research.

8.1 Retrieval model independence

The first question addressed in this thesis is how can we define an algebra, which is a formal mathematical specification, to abstract away from the retrieval model definition. To give an answer to this question we analyzed the problem of structured information retrieval, i.e., we analyzed different aspects of structured user queries. The result of this analysis is the identification of the four elementary structured retrieval requirements (see Section 3.1):

- *entity selection* – the selection of different entities in structured documents, such as elements, terms, attributes, image and video references, which are parts of the user query;
- *relevance score computation* – the computation of relevance scores for different structured elements with respect to the content they contain;
- *relevance score combination* – the combination of relevance scores from (different) elements in a document structure, resulting in a common element relevance score;
- *relevance score propagation* – the propagation of scores from different elements to common ancestor or descendant elements following the query.

Score region algebra is developed following these requirements (see Section 3.2). SRA models structured documents as a set of regions, where each region can represent different entities in structured documents. Regions also depict the score of each entity, determined with respect to the user query. On the domain of region sets, region operators are defined. The operators model different aspects of elementary structured retrieval requirements. Therefore, SRA consists of selection operators, score computation operators, score combination operators, and score propagation operators.

The algebra is the central part of a three-level database system developed for structured retrieval, called TIJAH. TIJAH supports the specification of user requests in a structured query language (NEXI), their mapping to an SRA query plans, and also the transformation from SRA operators to a physical (MIL) query plan and its execution within a database kernel (MonetDB). This is explained in Section 4.1.

Contributions

The main contributions of our approach presented in this thesis, with respect to supporting the instantiation of various retrieval models in the framework, are the following.

- *SRA enables retrieval model independence by abstracting away from the retrieval model specification.*

- *SRA as the central component of the three-level database system (TIJAH) enables transparent instantiation of retrieval models.*

To achieve retrieval model independence, retrieval models in SRA are defined through abstract scoring functions and abstract operators. In other words, all the region attributes, except the region relevance score attribute, are specified in the SRA operator definition. The region relevance score attribute values are computed using the information from regions in the (left and right) operand region sets through the abstract scoring functions. Therefore, the user/application program does not have to be aware of how the retrieval model is implemented, i.e., how these abstract scoring functions are instantiated.

In the three-level TIJAH architecture this abstraction from retrieval models is supported through a retrieval model dictionary, responsible for selecting which of the abstract scoring function instantiations will be used in the query execution. Based on the specification of abstract functions and the selection of retrieval model options in the retrieval model dictionary, the proper operators are instantiated at the physical level.

Having such a transparent architecture, the implementation of new retrieval models is fairly easy. One just needs to enable new options in the retrieval model data dictionary, to provide the formal specification of these retrieval models using SRA abstract functions and abstract operators, and to implement these models at the physical level reusing parts of already implemented ones. Therefore, TIJAH is a suitable platform for implementing and testing retrieval models on different structured retrieval tasks.

Future research

TIJAH uses the 'fairly simple' NEXI query language which does not support proximity search, specification of the importance of query parts, parent/child element selections, following/sibling element specifications, etc. Therefore, more advanced query languages can be specified on top of SRA, such as TexQuery (XQuery full-text extension) [7]. Using such query languages requires SRA data model extensions and operator set extensions. This is achievable in SRA as pointed out in this thesis (see Section 3.3 and Chapter 7).

The current implementation of the SRA operators is done in MonetDB using MonetDB interpreter language (MIL) [19]. MonetDB operators support database-like search. It is not always straightforward to implement structured IR operations (in MIL) using database operators. Therefore, new procedures have been added to the MonetDB kernel to support IR operations. It might be more appropriate to develop special-purpose physical implementations that would enable fast execution of IR operators. Another speed-up solution is the integration with the currently fastest XQuery engine implemented in MonetDB (called Pathfinder) [20]. First steps in this direction have already been taken [99].

However, optimization does not only depend on the execution times of physical operators, but also on the logical query plan. In Section 4.3 we studied the SRA operator properties. The goal of studying SRA operator properties is to enable logical query optimization, such as relational algebra query optimization, as can be seen in, e.g., [16], and obtaining the gain in efficiency. This thesis shows that the potential for such SRA query optimization exists. However, it demands further analysis of the physical implementation to estimate the cost of the query execution, especially when using operators implementing different scoring functions. We expect this to be an interesting question for future research.

8.2 Effectiveness of structured retrieval models

The main goal of any information retrieval model (system) is to be effective on a range of information retrieval tasks. For flat text IR numerous retrieval models exist that achieve high effectiveness, such as language models, BM25, and vector space models. However, structured information retrieval is in its infancy and every year new models are being developed achieving high effectiveness (see e.g., [74]). These new models are mostly based on flat text retrieval models adapted to structured retrieval and only a few of them have been developed specially for structured retrieval (see Section 2.1).

Furthermore, the structured retrieval task consists of many subtasks not recognized in flat text IR. For example, a subtask where a user does not specify the desired element to be returned but only the list of search terms. It is up to the retrieval system to determine the most relevant elements. Another scenario is where the user specifies structured constraints in the query that need to be followed strictly or vaguely by the search system. This diversity resulted in the existence of different types of user requests but also in different retrieval models being more appropriate for different user requests (see Chapters 5 and 6).

We argue that structured retrieval models can be decomposed into four components that match the four elementary structured retrieval requirements. Therefore, the answer element relevance score can be determined using different models for entity (element and term) selection, element relevance score computation, relevance score combination, and relevance score propagation. We demonstrate this in Chapter 4 where we specified variants of entity selection, score computation and score combination models, based on the original Boolean, GPX, language model, Okapi, and tf.idf definitions, and proposed several approaches for modeling score propagation.

Contributions

Considering the implementations of different structured retrieval models within SRA on (document and document component) retrieval effectiveness, the contributions that can be found in this thesis are the following.

- *Using the SRA framework, different retrieval models can be easily implemented, and they can be analyzed and compared with respect to their effectiveness.*

- *Choosing the right composition of score computation, combination, and propagation abstract functions the effectiveness can be improved for both document component retrieval and document retrieval.*

To demonstrate the benefits of easy instantiation of different retrieval models within the SRA framework numerous experiments are presented in Chapter 5 and 6. Chapter 5 illustrates the influence of different instantiations of score computation, score combination, and score propagation functions on retrieval model effectiveness for document component retrieval. The document component retrieval effectiveness depends a lot on the chosen implementations of scoring functions as well as on their composition. For example, language model score computation is more effective in composition with score combinations implemented as product (AND) and sum (OR), while Okapi score computation is more effective in composition with score combination implemented as sum (both AND and OR).

Considering the score propagation, different retrieval models show distinct behavior for the upwards score propagation, while for the downwards score propagation weighted sum produces less effective results than simple sum of scores of containing elements. Among retrieval models, the GPX and language model based ones are more effective on document component retrieval than Okapi based models. Also, the GPX based models are more precision oriented, while language models are more recall oriented.

Furthermore, we show that there is a relation between the type and the complexity of user requests and the effectiveness of some structured retrieval models. This is especially the case for different implementations of score combination functions (see Section 5.5). By choosing the right composition of scoring functions, for queries with different complexity and for different types of user requests, we can achieve higher effectiveness.

Similarly, Chapter 6 discusses a set of experiments applied to document retrieval. Here we exploit the usage of query structure and document structure for improving the effectiveness of flat text retrieval models. The results show that with the clever formulation of structured queries, using document structure and annotated document content, we can achieve higher effectiveness than when using flat text queries. The effectiveness is improved using structured queries even in comparison to using long queries formed out of the topic description and topic narrative.

The best compositions of scoring functions are similar to ones for document component retrieval, except that the upwards score propagation function when using weighted sum is less effective than simple sum in all experiments (see Section 6.6). We also show that the structured smoothing is important when propagating scores in collections where documents have incomplete structure (such as TREC and CLEF). Furthermore, by comparing the effectiveness of different retrieval models, we can see that the Okapi based models are more robust to different score propagation implementations and give higher effectiveness than the language model and GPX based models on document retrieval.

Future research

The instantiation of retrieval models following the structured retrieval requirements is based on flat text retrieval model specifications. However, due to different term distributions in structured elements in contrast to documents, and due to the hierarchical organization of structured documents, new models geared toward structured retrieval might give higher effectiveness. The TIJAH architecture would be a perfect environment for discovering such new score computation, score combination, and especially score propagation functions. Eventually, this would lead to the development of new models that could achieve higher effectiveness on different structured retrieval tasks.

Furthermore, in the experiments described in this thesis the semantics of annotations present in structured documents, as well as the relative position of these annotations in a hierarchical document structure, was not used. This information can be utilized for, e.g., assigning a non-uniform element prior (default region score) to different elements [178]. This prior can also be learned by, e.g., analyzing user preferences. Furthermore, such user preferences, along with the data obtained from analyzing the relations between the retrieval models and query types, can be used for the automatic selection of parameters in the retrieval model dictionary.

Finally, we mostly tested retrieval models on a strict scenario where structured constraints need to be followed strictly. Similar experiments, probably with new implementations or extensions of scoring functions, can be performed on other structured retrieval scenarios, e.g., content-only search, vague search (see [145]). Furthermore, advanced models that describe non-textual content search can be included in the experiments, and also their tighter integration with the text search can be tested (see below).

8.3 Flexibility and extensibility

To satisfy various user information needs modern information retrieval systems need to cope with diverse user requests that can include not only textual search but also search on images, videos, songs, etc. The design of the TIJAH system and score region algebra was driven also by the goal to support these searches (see Section 1.2.3). Therefore, our structured IR system is flexible in supporting retrieval models in various real world areas, and also with respect to querying and storage of various data sources.

In this thesis we present how SRA (and TIJAH) can be extended to support additional algebra operators and additional attributes in the SRA data model. We also show how search in non-textual domains can be incorporated in the framework. In all these cases the basic functionality of SRA remains the same.

The SRA data model extension is explained in Section 7.1 on an example that uses the element level as an additional content information employed in structured retrieval. Element level information is used for specifying new retrieval models.

Section 7.2 explains how video search is performed in the TIJAH system. The video data is modeled through speech transcripts of a video material, encoded in XML. To support search in such XML data, new retrieval models have been developed that utilize the specific structure of videos, consisting of scenes and shots.

Finally, Section 7.3 explains how image search (search by example) can be achieved in TIJAH. Image content is described in terms of low-level video features that are matched with features from the sample image to find the most similar images. The image search is then combined with the text search.

Contributions

The SRA data model and operators are designed in such a way that they can easily be extended. Thus, the contributions presented in this thesis, with respect to distinct content information conveyed in structured documents are as follows.

- *SRA data model and operator set abstract away from the modeling of document content at the physical level, providing content description independence.*

- *The SRA data model can be extended for modeling additional information that describes different types of document content, without affecting the core SRA functionality.*

Besides enabling retrieval model independence and transparent instantiation of retrieval models, SRA (the TIJAH system) enables the independence between the generation of a query plan at the logical level and the storage structures and operators responsible for data manipulation at the physical level. However, this is not related only to how content is stored at the physical level (physical data independence) but also to the high-level representation of distinct types of document content used for defining retrieval models.

Having distinct selection operators, different types of document content (e.g., images, videos, songs) can be selected based on their similarity/equivalence with the reference content, and incorporated into the SRA framework without major changes. This is possible as SRA supports content description independence.

In such a way, arbitrary content can be modeled and accessed in SRA. Using abstract functions, the reasoning can be performed with respect to the relevance of such content to user queries. At the physical level of the TIJAH system textual, video, image, and audio content can be represented and stored in different ways, e.g., terms with and without stemming, video as a transcript of recognized speech (see Section 7.2), image as a set of high-level image features (see Section 7.3). The SRA does not have to deal with these internals but only with the way in which this data is used to compute the (matching) scores for the respective elements inside a document.

In Chapter 7 we demonstrate that special purpose retrieval models on a video retrieval task, that use specific structure of videos (each video consists of scenes, and each scene of shots), significantly improve the effectiveness in comparison to the baseline language model. However, in the experiments presented in Sections 7.1 and 7.3 we were not able to improve the retrieval effectiveness when using level information or simple image retrieval models for structured search. Nevertheless, some more advanced retrieval models might achieve this goal.

Future research

The SRA extensions presented in Chapter 7 illustrate how SRA can be adapted to different structured retrieval tasks. Many opportunities exist for improving either expressiveness of the SRA or effectiveness of retrieval models in textual, video, and image domain. For example, one approach to improve SRA expressiveness is to introduce new attributes and operators to support the XQuery full text (TeXQuery) query language. This might result in the integration of TIJAH with Pathfinder at the physical level (MonetDB) as already explained in the part that discusses future research in Section 8.1.

Furthermore, we can upgrade the retrieval models to use term proximity for relevance computation, or semantics of document annotations when computing, combining, and propagating scores. Maybe some more advanced phrase search methods can improve the results. These are just some of the possible extensions toward the development of a more effective structured retrieval system.

The image similarity search models employed in Section 7.3 did not show any significant improvements over models that use image caption text and textual description for image search. However, the usage of more advanced image search techniques might give better results. Similarly for video retrieval, audio and video content analysis techniques can be used to achieve better description of video sequences. Also more advanced image similarity search techniques can be used for enhancing the video search subpart of the TIJAH system.

Appendix A

INEX NEXI queries

A.1 2003 content-and-structure queries

```
61. //article[about(.,clustering +distributed) and about(.//sec,java)]
62. //article[about(.,security +biometrics)
       AND about(.//sec,"facial recognition")]
63. //article[about(.,"digital library")
       AND about(.//p, +authorization +"access control" +security)]
64. //article[about(., hollerith)]//sec[about(., dehomag)]
65. //article[.//fm//yr > 1998 AND about(., "image retrieval")]
66. //article[.//fm//yr < 2000]//sec[about(.,"search engines")]
67. //article//fm[about(.//(tig|abs), +software +architecture)
       and about(., -distributed -web)]
68. //article[about(., +smalltalk) or about(., +lisp) or about(.,+erlang)
       or about(., +java)]
       //bdy//sec[about(., +"garbage collection" +algorithm)]
69. //article//bdy//sec[about(.//st,"information retrieval")]
70. //article[about(.//fm//abs, "information retrieval"
          "digital libraries")]
71. //article[about(.,formal methods verify correctness aviation systems)]
       //bdy//*[about(.,case study application model checking theorem
          proving)]
72. //article[about(.//fm//au//aff,united states of america)]
       //bdy//*[about(.,weather forecasting systems)]
73. //article[about(.//st,+comparison) and
       about(.//bib,"machine learning")]
74. //article[about(., video streaming applications)]
       //sec[about(., media stream synchronization)
       OR about(., stream delivery protocol)]
75. //article[about(., petri net) AND about(.//sec, formal definition)
       AND about(.//sec, algorithm efficiency computation approximation)]
76. //article[(.//fm//yr = 2000 OR .//fm//yr = 1999)
       AND about(., "intelligent transportation system")]
       //sec[about(.,automation +vehicle)]
77. //article[about(.//sec,"reverse engineering")]//sec[about(., legal)
       OR about(.,legislation)]
78. //vt[about(.,"information retrieval" student)]
79. //article[about(.,xml) AND about(.,database)]
80. //article//bdy//sec[about(.,"clock synchronization"
          "distributed systems")]
81. //article[about(.//p,"multi concurrency control") AND
       about(.//p, algorithm) AND about(.//fm//atl, databases)]
82. //article[about(.,handwriting recognition) AND about(.//fm//au,kim)]
83. //article//fm//abs[about(., "data mining" "frequent itemset")]
84. //p[about(.,overview "distributed query processing" join)]
85. //article[.//fm//yr >= 1998 and .//fig//no > 9]//sec[about(.//p, VR
          "virtual reality" "virtual environment" cyberspace
          "augmented reality")]
86. //sec[about(.,mobile electronic payment system)]
87. //article[(.//fm//yr = 1998 OR .//fm//yr = 1999 OR .//fm//yr = 2000
       OR .//fm//yr = 2001 OR .//fm//yr = 2002)
       AND about(., "support vector machines")]
88. //article[(.//fm//yr = 1998 OR .//fm//yr = 1999 OR .//fm//yr = 2000
       OR .//fm//yr = 2001) AND about(., "web crawler")]
89. //article[about(.//bdy,clustering "vector quantization" +fuzzy +k-means
          +c-means -sofm -som)]//bm//bb[about(.,"vector quantization"
          +fuzzy clustering +k-means +c-means) AND about(.//pdt,1999)
       AND about(.//au//snm, -kohonen)]
90. //article[about(.//sec,+trust authentication "electronic commerce"
          e-commerce ebusiness marketplace)]
       //abs[about(., trust authentication)]
```

A.2 2004 content-and-structure queries

```
127. //sec//(p|fgc)[about( ., godel lukasiewicz and other fuzzy
        implication definitions)]
128. //article[about(., intelligent transport systems)]
        //sec[about(., on-board route planning navigation system for
           automobiles)]
129. //article[about(.//atl, new book review bookshelf)]
        //sec[about(., database "data warehouse")]
130. //article[about(.//p,object database)]
        //p[about(.,version management)]
131. //article[about(.//au,"jiawei han")]//abs[about(.,"data mining")]
132. //article[about(.//abs,classification)]
        //sec[about(.,experiment compare)]
133. //article[about(.//fm//tig//atl, query)
        and about(.//st, optimization)]
134. //article[(about(., "phrase search") OR about(., "proximity search")
        OR about(., "string matching")) AND (about(.,tries)
        OR about(.,"suffix trees") OR about(.,"pat arrays"))]
        //sec[about(.,algorithm)]
135. //article[about(.//atl, summaries)]
```

```
        //sec[about(., "Internet security") or
        about(.,"network security")]
136. //bib[about(., text categorisation) and
        about(., "support vector machines" svm)]
137. //article [about(.//abs, "digital library") or
        about(.//ip1, "digital library")]
138. //article[about(.,operating system) and
        about(.//sec,thread implementation)]
139. //article[(about(.//bb//au//snm, bertino) or
        about(.//bb//au//snm , jajodia)) and
        about(.//bb//atl, security model) and
        about(.//bb//atl, -"indexing model" -"object oriented model")]
140. //article[about(., xml)]//bdy//sec[about(., "information integration"
           +web) or about(., "information exchange" +web)]
141. //article[about(.,java)]//sec[about(.,implementing threads)]
142. //abs[about(.,database access using perl)]
143. //sec[about(.,+stemming +information)]
144. //article[about(.//abs, bioinformatics "software systems") and
         about(.//p,data warehouse multi-format multi-type integration)]
145. //article[about(.,information retrieval)]
         //p[about(.,relevance feedback)]
146. //article[(.//fm//yr > 1999)]//sec[about(.//*, xml html web)]
147. //sec[about(.//st,conclusions) AND about(.,web) AND about(.,distance)
        AND about (.,learning)]
149. //article[about(.//(abs|kwd), "genetic algorithm")]
        //bdy//sec[about(.,simulated annealing)]
150. //article[about(., animation)]
        //bdy//sec[about(.//st, inverse kinematics)]
151. //article[about(., "web search engine")]
        //sec[about(., "vector space model")]
152. //article//p[about(.,+linux "word processing" "word processor"
        "office programs")]
153. //article//bm//vt[about(.,"phd student") OR about(.,"phd candidate")]
154. //article[about(.//bib, abiteboul)]
        //bdy//*[about(., semistructured + query)]
155. //article[about(.//p,"self organising feature map") and
        about(.//fm//yr,2000)]//fig[about(.//fgc,"self organising map")]
156. //article[about(.//abs, "spatial join")]
        //bdy//sec[about(., "performance evaluation")]
157. //article[about(.//abs,-query -"query optimization" -linear)
        and about(.//bdy,newton +gradient hessian technique)]
        //bdy//*[about(.,+optimization -experiments)
        and (about(.,maximization) or about(.,minimization))]
158. //article[about(.//fm, turing test) or about(.//abs, turing test)]
        //bdy[about(., turning test consciousness intelligence imitation
           game)]
159. //article[about(.,"bayesian networks")]
```

```
        //sec[about(.,learning structure)]
160. //article[about(., image retrieval)]
        //sec[about(., "latent semantic indexing")]
161. //article[about(., database access methods for spatial data and text)]
        //bm//bb[about(.//atl, database access methods)]
```

Appendix B

NEXI version of TREC queries

B.1 Title queries

```
301. //DOC[about(., international organized crime)]
302. //DOC[about(., poliomyelitis and post polio)]
303. //DOC[about(., hubble telescope achievements)]
304. //DOC[about(., endangered species mammals)]
305. //DOC[about(., most dangerous vehicles)]
306. //DOC[about(., african civilian deaths)]
307. //DOC[about(., new hydroelectric projects)]
308. //DOC[about(., implant dentistry)]
309. //DOC[about(., rap and crime)]
310. //DOC[about(., radio waves and brain cancer)]
311. //DOC[about(., industrial espionage)]
312. //DOC[about(., hydroponics)]
313. //DOC[about(., magnetic levitation maglev)]
314. //DOC[about(., marine vegetation)]
315. //DOC[about(., unexplained highway accidents)]
316. //DOC[about(., polygamy polyandry polygyny)]
317. //DOC[about(., unsolicited faxes)]
318. //DOC[about(., best retirement country)]
319. //DOC[about(., new fuel sources)]
320. //DOC[about(., undersea fiber optic cable)]
321. //DOC[about(., women in parliaments)]
322. //DOC[about(., international art crime)]
323. //DOC[about(., literary journalistic plagiarism)]
324. //DOC[about(., argentine british relations)]
325. //DOC[about(., cult lifestyles)]
326. //DOC[about(., ferry sinkings)]
327. //DOC[about(., modern slavery)]
328. //DOC[about(., pope beatifications)]
329. //DOC[about(., mexican air pollution)]
330. //DOC[about(., iran iraq cooperation)]
331. //DOC[about(., world bank criticism)]
332. //DOC[about(., income tax evasion)]
333. //DOC[about(., antibiotics bacteria disease)]
```

```
334. //DOC[about(., export controls cryptography)]
335. //DOC[about(., adoptive biological parents)]
336. //DOC[about(., black bear attacks)]
337. //DOC[about(., viral hepatitis)]
338. //DOC[about(., risk of aspirin)]
339. //DOC[about(., alzheimer drug treatment)]
340. //DOC[about(., land mine ban)]
341. //DOC[about(., airport security)]
342. //DOC[about(., diplomatic expulsion)]
343. //DOC[about(., police deaths)]
344. //DOC[about(., abuses of e mail)]
345. //DOC[about(., overseas tobacco sales)]
346. //DOC[about(., educational standards)]
347. //DOC[about(., wildlife extinction)]
348. //DOC[about(., agoraphobia)]
349. //DOC[about(., metabolism)]
350. //DOC[about(., health and computer terminals)]
```

B.2 Expanded queries

```
301. //DOC[about(., trade traffic cocaine drugs cartel organizations
        organisations criminal crime)]
302. //DOC[about(., poliomyelitis post polio control protection)]
303. //DOC[about(., hubble telescope achievements accomplishments
        knowledge hypotheses hypothesis)]
304. //DOC[about(., endangered species mammals)]
305. //DOC[about(., vehicles cars crash crashworthy death danger)]
306. //DOC[about(., civil civilian war africa african death casualties)]
307. //DOC[about(., new proposed planned hydroelectric projects)]
308. //DOC[about(., implants tooth teeth dentistry dentist)]
309. //DOC[about(., rap crime violence drugs suicide teenagers youth
        young)]
310. //DOC[about(., radio waves brain cancer tumor tumour)]
311. //DOC[about(., industrial espionage theft thievery trade secrets)]
312. //DOC[about(., hydroponics advantages agricultural nutrients
        substrates)]
313. //DOC[about(., magnetic levitation maglev)]
314. //DOC[about(., food drug medicine algae seaweed kelp marine
        vegetation)]
315. //DOC[about(., highway accidents unresolved unexplained)]
316. //DOC[about(., prevalence polygamy -serial)]
317. //DOC[about(., unsolicited junk fax faxes faxing facsimile cost
        laws regulation privacy)]
318. //DOC[about(., retirement retiree living conditions foreign abroad)]
319. //DOC[about(., research fuel sources)]
```

```
320. //DOC[about(., undersea underwater fiber optic cable link)]
321. //DOC[about(., women parliaments representation legislatures)]
322. //DOC[about(., international art trade embezzlement fraud)]
323. //DOC[about(., plagiarism literary journalistic journalism)]
324. //DOC[about(., argentine british international relations exchanges)]
325. //DOC[about(., cults activities lifestyles organization dress)]
326. //DOC[about(., ferry sinking sunk death casualties)]
327. //DOC[about(., slaves slavery present modern today buy sell commerce)]
328. //DOC[about(., pope beatified beatifications)]
329. //DOC[about(., mexico city air pollution)]
330. //DOC[about(., iran iraq cooperation friend friendly friendship
        relations)]
331. //DOC[about(., world bank  criticisms accusations unfair unfairly)]
332. //DOC[about(., unitedstates us usa tax evasion investigations)]
333. //DOC[about(., bacteria antibiotics)]
334. //DOC[about(., export controls encryption cryptography)]
335. //DOC[about(., biological parents adopt adoption adopted children
        child)]
336. //DOC[about(., black bear attacks maul mauled mauling)]
337. //DOC[about(., viral hepatitis progress treatment research medical
        medicin vaccines drug)]
338. //DOC[about(., aspirin adverse risks)]
339. //DOC[about(., alzheimer treatment prevention drug medecine)]
340. //DOC[about(., ban land mines)]
341. //DOC[about(., airport security passengers luggage carry-on)]
342. //DOC[about(., diplomatic expulsion information sensitive secret
        classified trade technology industrial)]
343. //DOC[about(., police policeman policewoman policemen policewomen
        death killed shot shooting trial witness testify)]
344. //DOC[about(., email e electronic mail abuse spam)]
345. //DOC[about(., tobacco cigarette sales abroad overseas foreign)]
346. //DOC[about(., education standards abroad overseas foreign)]
347. //DOC[about(., wildlife animals species extinction disappear)]
348. //DOC[about(., agoraphobia agoraphobic agora phobia)]
349. //DOC[about(., metabolic metabolism catabolic anabolic glycolysis
        krebs)]
350. //DOC[about(., carpel tunnel rsi cataracts fatigue
        disordercomputers)]
```

B.3 Faceted queries

```
301. //DOC[(about(., trade) OR about(., traffic)) AND (about(., cocaine)
        OR about(., drugs)) AND (about(., cartel) OR
        about(., organizations) OR about(., organisations)) AND
        (about(., criminal) OR about(., crime))]
```

```
302. //DOC[(about(., poliomyelitis) OR about(., post polio) OR
       about(., polio)) AND (about(., control) OR about(., protection))]
303. //DOC[about(., hubble telescope) AND (about(., achievements) OR
       about(., accomplishments)) AND (about(., knowledge) OR
       about(., hypotheses) OR about(., hypothesis))]
304. //DOC[about(., endangered species mammals)]
305. //DOC[(about(., vehicles) OR about(., cars)) AND (about(., crash) OR
       about(., crashworthy) OR about(., death) OR about(., danger))]
306. //DOC[about(., civil civilian war) AND (about(., africa) OR
       about(., african)) AND (about(., death) OR about(., casualties))]
307. //DOC[(about(., new) OR about(., proposed) OR about(., planned)) AND
       about(., hydroelectric projects)]
308. //DOC[about(., implants) AND (about(., tooth) OR about(., teeth) OR
       about(., dentistry) OR about(., dentist))]
309. //DOC[about(., rap) AND (about(., crime) OR about(., violence) OR
       about(., drugs) OR about(., suicide)) AND (about(., teenagers) OR
       about(., youth) OR about(., young))]
310. //DOC[about(., radio waves brain) AND (about(., cancer) OR
       about(., tumor) OR about(., tumour))]
311. //DOC[about(., industrial espionage) OR ((about(., theft) OR
       about(., thievery)) AND about(., trade secrets))]
312. //DOC[about(., hydroponics advantages agricultural) AND
       (about(., nutrients) OR about(., substrates))]
313. //DOC[about(., magnetic levitation) OR about(., maglev)]
314. //DOC[(about(., food) OR about(., drug) OR about(., medicine)) AND
       (about(., algae) OR about(., seaweed) OR about(., kelp) OR
       about(., marine vegetation))]
315. //DOC[about(., highway accidents) AND (about(., unresolved) OR
       about(., unexplained))]
316. //DOC[about(., prevalence polygamy -serial)]
317. //DOC[(about(., unsolicited) OR about(., junk)) AND (about(., fax) OR
       about(., faxes) OR about(., faxing) OR about(., facsimile)) AND
       (about(., cost) OR about(., laws) OR about(., regulation) OR
       about(., privacy))]
318. //DOC[(about(., retirement) OR about(., retiree)) AND
       about(., living conditions) AND (about(., foreign) OR
       about(., abroad))]
319. //DOC[about(., research fuel sources)]
320. //DOC[(about(., undersea) OR about(., underwater)) AND
       about(., fiber optic) AND (about(., cable) OR about(., link))]
321. //DOC[about(., women) AND (about(., parliaments) OR
       about(., representation) OR about(., legislatures))]
322. //DOC[about(., international art trade) AND (about(., embezzlement)
       OR about(., fraud))]
323. //DOC[about(., plagiarism) AND (about(., literary) OR
       about(., journalistic) OR about(., journalism))]
324. //DOC[about(., argentine british) AND (about(., international
          relations) OR about(., exchanges))]
325. //DOC[about(., cults) AND (about(., activities) OR
        about(., lifestyles) OR about(., organization) OR about(., dress))]
326. //DOC[about(., ferry) AND (about(., sinking) OR about(., sunk)) AND
        (about(., death) OR about(., casualties))]
327. //DOC[(about(., slaves) OR about(., slavery)) AND (about(., present)
        OR about(., modern) OR about(., today)) AND (about(., buy)
        OR about(., sell) OR about(., commerce))]
328. //DOC[about(., pope) AND (about(., beatified) OR
        about(., beatifications))]
329. //DOC[about(., mexico city air pollution)]
330. //DOC[about(., iran iraq) AND (about(., cooperation) OR
        (about(., friend) OR about(., friendly) OR about(., friendship)
        AND about(., relations)))]
331. //DOC[about(., world bank) AND (about(., criticisms) OR
        about(., accusations) OR about(., unfair) OR about(., unfairly))]
332. //DOC[(about(., united states) OR about(., us) OR about(., usa)) AND
        about(., tax evasion investigations)]
333. //DOC[about(., bacteria antibiotics)]
334. //DOC[about(., export controls) AND (about(., encryption) OR
        about(., cryptography))]
335. //DOC[about(., biological parents) AND (about(., adopt) OR
        about(., adoption) OR about(., adopted)) AND (about(., children)
        OR about(., child))]
336. //DOC[about(., black bear) AND (about(., attacks) OR about(., maul)
        OR about(., mauled) OR about(., mauling))]
337. //DOC[about(., viral hepatitis) AND (about(., progress) OR
        about(., treatment) OR about(., research) OR about(., medical) OR
        about(., medicin) OR about(., vaccines) OR about(., drug))]
338. //DOC[about(., aspirin) AND (about(., adverse) OR about(., risks))]
339. //DOC[about(., alzheimer) AND (about(., treatment) OR
        about(., prevention)) AND (about(., drug) OR about(., medecine))]
340. //DOC[about(., ban land mines)]
341. //DOC[about(., airport security) AND (about(., passengers) OR
        about(., luggage) OR about(., carry on))]
342. //DOC[about(., diplomatic expulsion information) AND
        (about(., sensitive) OR about(., secret) OR about(., classified))
        AND (about(., trade) OR about(., technology) OR
        about(., industrial))]
343. //DOC[(about(., police) OR about(., policeman) OR
        about(., policewoman) OR about(., policemen) OR
        about(., policewomen)) AND (about(., death) OR about(., killed)
        OR about(., shot) OR about(., shooting)) AND (about(., trial) OR
        about(., witness) OR about(., testify))]
344. //DOC[(about(., email) OR about(., e mail) OR
        about(., electronic mail)) AND (about(., abuse) OR
        about(., spam))]
```

```
345. //DOC[(about(., tobacco) OR about(., cigarette)) AND
        about(., sales) AND (about(., abroad) OR about(., overseas) OR
        about(., foreign))]
346. //DOC[about(., education standards) AND (about(., abroad) OR
        about(., overseas) OR about(., foreign))]
347. //DOC[about(., wildlife) AND (about(., animals) OR about(., species))
        AND (about(., extinction) OR about(., disappear))]
348. //DOC[about(., agoraphobia) OR about(., agoraphobic) OR
        about(., agora phobia)]
349. //DOC[(about(., metabolic) OR about(., metabolism)) AND
        (about(., catabolic) OR about(., anabolic) OR
        about(., glycolysis) OR about(., krebs))]
350. //DOC[(about(., carpel tunnel) OR about(., rsi) OR
        about(., cataracts) OR about(., fatigue) OR about(., disorder))
        AND about(., computers)]
```

B.4 Field-based + title queries

```
301. //DOC[about(., international organized crime) AND
        about(.//SUBJECT, drug crime)]
302. //DOC[about(., poliomyelitis and post polio) AND
        about(.//SUBJECT, health)]
303. //DOC[about(., hubble telescope achievements) AND
        about(.//SUBJECT, space development)]
304. //DOC[about(., endangered species mammals) AND
        about(.//SUBJECT, animal)]
305. //DOC[about(., most dangerous vehicles) AND
        about(.//SUBJECT, safety automobile accidents)]
306. //DOC[about(., african civilian deaths)]
307. //DOC[about(., new hydroelectric projects)]
308. //DOC[about(., implant dentistry) AND
        about(.//SUBJECT, medical health)]
309. //DOC[about(., rap and crime) AND
        about(.//SUBJECT, accidents crime weapons shootings)]
310. //DOC[about(., radio waves and brain cancer) AND
        about(.//SUBJECT, health)]
311. //DOC[about(., industrial espionage) AND
        about(.//SUBJECT, industry)]
312. //DOC[about(., hydroponics)]
313. //DOC[about(., magnetic levitation maglev) AND
        about(.//SUBJECT, transportation)]
314. //DOC[about(., marine vegetation) AND
        about(.//SUBJECT, agriculture)]
315. //DOC[about(., unexplained highway accidents) AND
```

```
        about(.//SUBJECT, accidents traffic)]
316. //DOC[about(., polygamy polyandry polygyny) AND
        about(.//SUBJECT, etics law)]
317. //DOC[about(., unsolicited faxes) AND
        about(.//SUBJECT, economy)]
318. //DOC[about(., best retirement country)]
319. //DOC[about(., new fuel sources)]
320. //DOC[about(., undersea fiber optic cable)]
321. //DOC[about(., women in parliaments) AND
        about(.//SUBJECT, politics)]
322. //DOC[about(., international art crime) AND
        about(.//SUBJECT, fraud sales robberies)]
323. //DOC[about(., literary journalistic plagiarism) AND
        about(.//SUBJECT, rights)]
324. //DOC[about(., argentine british relations) AND
        about(.//SUBJECT, foreign relations international)]
325. //DOC[about(., cult lifestyles) AND
        about(.//SUBJECT, culture)]
326. //DOC[about(., ferry sinkings) AND
        about(.//SUBJECT, accidents traffic)]
327. //DOC[about(., modern slavery)]
328. //DOC[about(., pope beatifications)]
329. //DOC[about(., mexican air pollution) AND
        about(.//SUBJECT, pollution)]
330. //DOC[about(., iran iraq cooperation) AND
        about(.//SUBJECT, international)]
331. //DOC[about(., world bank criticism) AND
        about(.//SUBJECT, economy finances)]
332. //DOC[about(., income tax evasion) AND
        about(.//SUBJECT, fraud)]
333. //DOC[about(., antibiotics bacteria disease) AND
        about(.//SUBJECT, medical health)]
334. //DOC[about(., export controls cryptography) AND
        about(.//SUBJECT, laws computers legislation safety government)]
335. //DOC[about(., adoptive biological parents) AND
        about(.//SUBJECT, rights)]
336. //DOC[about(., black bear attacks) AND
        about(.//SUBJECT, animal injuries)]
337. //DOC[about(., viral hepatitis) AND
        about(.//SUBJECT, health)]
338. //DOC[about(., risk of aspirin) AND
        about(.//SUBJECT, medical drug health)]
339. //DOC[about(., alzheimer drug treatment) AND
        about(.//SUBJECT, medical drug)]
340. //DOC[about(., land mine ban) AND
        about(.//SUBJECT, foreign relations military)]
341. //DOC[about(., airport security) AND
```

```
        about(.//SUBJECT, security)]
342. //DOC[about(., diplomatic expulsion) AND
        about(.//SUBJECT, government)]
343. //DOC[about(., police deaths) AND
        about(.//SUBJECT, police accident murders)]
344. //DOC[about(., abuses of e mail) AND
        about(.//SUBJECT, computer)]
345. //DOC[about(., overseas tobacco sales) AND
        about(.//SUBJECT, sales industry)]
346. //DOC[about(., educational standards) AND
        about(.//SUBJECT, education school college)]
347. //DOC[about(., wildlife extinction) AND
        about(.//SUBJECT, animals)]
348. //DOC[about(., agoraphobia) AND
        about(.//SUBJECT, research)]
349. //DOC[about(., metabolism)]
350. //DOC[about(., health and computer terminals) AND
        about(.//SUBJECT, health computer)]
```

B.5 Field-based + faceted queries

```
301. //DOC[(about(., trade) OR about(., traffic)) AND (about(., cocaine)
        OR about(., drugs)) AND (about(., cartel) OR
        about(., organizations) OR about(., organisations)) AND
        (about(., criminal) OR about(., crime)) AND
        about(.//SUBJECT, drug crime)]
302. //DOC[(about(., poliomyelitis) OR about(., post polio) OR
        about(., polio)) AND (about(., control) OR about(., protection))
        AND about(.//SUBJECT, health)]
303. //DOC[about(., hubble telescope) AND (about(., achievements) OR
        about(., accomplishments)) AND (about(., knowledge) OR
        about(., hypotheses) OR about(., hypothesis)) AND
        about(.//SUBJECT, space development)]
304. //DOC[about(., endangered species mammals) AND
        about(.//SUBJECT, animal)]
305. //DOC[(about(., vehicles) OR about(., cars)) AND (about(., crash) OR
        about(., crashworthy) OR about(., death) OR about(., danger)) AND
        about(.//SUBJECT, safety automobile accidents)]
306. //DOC[about(., civil civilian war) AND (about(., africa) OR
        about(., african)) AND (about(., death) OR about(., casualties))]
307. //DOC[(about(., new) OR about(., proposed) OR about(., planned)) AND
        about(., hydroelectric projects)]
308. //DOC[about(., implants) AND (about(., tooth) OR about(., teeth) OR
        about(., dentistry) OR about(., dentist)) AND
        about(.//SUBJECT, medical health)]]
309. //DOC[about(., rap) AND (about(., crime) OR about(., violence) OR
        about(., drugs) OR about(., suicide)) AND (about(., teenagers) OR
        about(., youth) OR about(., young)) AND
        about(.//SUBJECT, accidents crime weapons shootings)]
310. //DOC[about(., radio waves brain) AND (about(., cancer) OR
        about(., tumor) OR about(., tumour)) AND
        about(.//SUBJECT, health)]
311. //DOC[about(., industrial espionage) OR ((about(., theft) OR
        about(., thievery)) AND about(., trade secrets)) AND
        about(.//SUBJECT, industry)]
312. //DOC[about(., hydroponics advantages agricultural) AND
        (about(., nutrients) OR about(., substrates))]
313. //DOC[about(., magnetic levitation) OR about(., maglev) AND
        about(.//SUBJECT, transportation)]
314. //DOC[(about(., food) OR about(., drug) OR about(., medicine)) AND
        (about(., algae) OR about(., seaweed) OR about(., kelp) OR
        about(., marine vegetation)) AND about(.//SUBJECT, agriculture)]
315. //DOC[about(., highway accidents) AND (about(., unresolved) OR
        about(., unexplained)) AND about(.//SUBJECT, accidents traffic)]
316. //DOC[about(., prevalence polygamy -serial) AND
        about(.//SUBJECT, etics law)]
317. //DOC[(about(., unsolicited) OR about(., junk)) AND (about(., fax) OR
        about(., faxes) OR about(., faxing) OR about(., facsimile)) AND
        (about(., cost) OR about(., laws) OR about(., regulation) OR
        about(., privacy)) AND about(.//SUBJECT, economy)]
318. //DOC[(about(., retirement) OR about(., retiree)) AND
        about(., living conditions) AND (about(., foreign) OR
        about(., abroad))]
319. //DOC[about(., research fuel sources)]
320. //DOC[(about(., undersea) OR about(., underwater)) AND
        about(., fiber optic) AND (about(., cable) OR about(., link))]
321. //DOC[about(., women) AND (about(., parliaments) OR
        about(., representation) OR about(., legislatures)) AND
        about(.//SUBJECT, politics)]
322. //DOC[about(., international art trade) AND (about(., embezzlement) OR
        about(., fraud)) AND about(.//SUBJECT, fraud sales robberies)]
323. //DOC[about(., plagiarism) AND (about(., literary) OR
        about(., journalistic) OR about(., journalism)) AND
        about(.//SUBJECT, rights)]
324. //DOC[about(., argentine british) AND
        (about(., international relations)
        OR about(., exchanges)) AND
        about(.//SUBJECT, foreign relations international)]
325. //DOC[about(., cults) AND (about(., activities) OR
        about(., lifestyles) OR about(., organization) OR about(., dress))
        AND about(.//SUBJECT, culture)]
```

```
326. //DOC[about(., ferry) AND (about(., sinking) OR about(., sunk)) AND
        (about(., death) OR about(., casualties)) AND
        about(.//SUBJECT, accidents traffic)]
327. //DOC[(about(., slaves) OR about(., slavery)) AND (about(., present)
        OR about(., modern) OR about(., today)) AND (about(., buy)
        OR about(., sell) OR about(., commerce))]
328. //DOC[about(., pope) AND (about(., beatified) OR
        about(., beatifications))]
329. //DOC[about(., mexico city air pollution) AND
        about(.//SUBJECT, pollution)]
330. //DOC[about(., iran iraq) AND (about(., cooperation) OR
        (about(., friend) OR about(., friendly) OR about(., friendship) AND
        about(., relations))) AND about(.//SUBJECT, international)]
331. //DOC[about(., world bank) AND (about(., criticisms) OR
        about(., accusations) OR about(., unfair) OR about(., unfairly))
        AND about(.//SUBJECT, economy finances)]
332. //DOC[(about(., united states) OR about(., us) OR about(., usa)) AND
        about(., tax evasion investigations) AND about(.//SUBJECT, fraud)]
333. //DOC[about(., bacteria antibiotics) AND
        about(.//SUBJECT, medical health)]
334. //DOC[about(., export controls) AND (about(., encryption) OR
        about(., cryptography)) AND
        about(.//SUBJECT, laws computers legislation safety government)]
335. //DOC[about(., biological parents) AND (about(., adopt) OR
        about(., adoption) OR about(., adopted)) AND (about(., children) OR
        about(., child)) AND about(.//SUBJECT, rights)]
336. //DOC[about(., black bear) AND (about(., attacks) OR about(., maul) OR
        about(., mauled) OR about(., mauling)) AND
        about(.//SUBJECT, animal injuries)]
337. //DOC[about(., viral hepatitis) AND (about(., progress) OR
        about(., treatment) OR about(., research) OR about(., medical) OR
        about(., medicin) OR about(., vaccines) OR about(., drug)) AND
        about(.//SUBJECT, health)]
338. //DOC[about(., aspirin) AND (about(., adverse) OR about(., risks))
        AND about(.//SUBJECT, medical drug health)]
339. //DOC[about(., alzheimer) AND (about(., treatment) OR
        about(., prevention)) AND (about(., drug) OR about(., medecine))
        AND about(.//SUBJECT, medical drug)]
340. //DOC[about(., ban land mines) AND
about(.//SUBJECT, foreign relations military)]
341. //DOC[about(., airport security) AND (about(., passengers) OR
        about(., luggage) OR about(., carry on)) AND
        about(.//SUBJECT, security)]
342. //DOC[about(., diplomatic expulsion information) AND
        (about(., sensitive) OR about(., secret) OR about(., classified))
        AND (about(., trade) OR about(., technology) OR
        about(., industrial)) AND about(.//SUBJECT, government)]
```

```
343. //DOC[(about(., police) OR about(., policeman) OR
        about(., policewoman) OR about(., policemen) OR
        about(., policewomen)) AND (about(., death) OR about(., killed)
        OR about(., shot) OR about(., shooting)) AND (about(., trial) OR
        about(., witness) OR about(., testify)) AND
        about(.//SUBJECT, police accident murders)]
344. //DOC[(about(., email) OR about(., e mail) OR
        about(., electronic mail)) AND (about(., abuse) OR
        about(., spam)) AND about(.//SUBJECT, computer)]
345. //DOC[(about(., tobacco) OR about(., cigarette)) AND
        about(., sales) AND (about(., abroad) OR about(., overseas) OR
        about(., foreign)) AND about(.//SUBJECT, sales industry)]
346. //DOC[about(., education standards) AND (about(., abroad) OR
        about(., overseas) OR about(., foreign)) AND
        about(.//SUBJECT, education school college)]
347. //DOC[about(., wildlife) AND (about(., animals) OR about(., species))
        AND (about(., extinction) OR about(., disappear)) AND
        about(.//SUBJECT, animals)]
348. //DOC[about(., agoraphobia) OR about(., agoraphobic) OR
        about(., agora phobia) AND about(.//SUBJECT, research)]
349. //DOC[(about(., metabolic) OR about(., metabolism)) AND
        (about(., catabolic) OR about(., anabolic) OR
        about(., glycolysis) OR about(., krebs))]
350. //DOC[(about(., carpel tunnel) OR about(., rsi) OR
        about(., cataracts) OR about(., fatigue) OR about(., disorder))
        AND about(., computers) AND about(.//SUBJECT, health computer)]
```

Appendix C

NEXI version of TRECVID queries

C.1 TRECVID 2003 queries

```
100. //VideoSegment[about(., building road city city suburbs)]
101. //VideoSegment[about(., basket basketball hoop net)]
102. //VideoSegment[about(., pitcher baseball game ball batter swings)]
103. //VideoSegment[about(., yasser arafat)]
104. //VideoSegment[about(., airplane taking off)]
105. //VideoSegment[about(., helicopter)]
106. //VideoSegment[about(., tomb unknown soldier arlington national
        cemetery)]
107. //VideoSegment[about(., rocket missile taking off)]
108. //VideoSegment[about(., mercedes logo)]
109. //VideoSegment[about(., tanks)]
110. //VideoSegment[about(., person diving water)]
111. //VideoSegment[about(., locomotive railroad cars)]
112. //VideoSegment[about(., flames fire)]
113. //VideoSegment[about(., snow covered moutain peaks ridges
        mountains)]
114. //VideoSegment[about(., osama bin laden )]
115. //VideoSegment[about(., roads vehicles traffic highway)]
116. //VideoSegment[about(., sphinx)]
117. //VideoSegment[about(., people crowd walking streets traffic
        buildings city street car)]
118. //VideoSegment[about(., congressman mark souder)]
119. //VideoSegment[about(., morgan freeman)]
120. //VideoSegment[about(., dow jones industrial average rise day
        points)]
121. //VideoSegment[about(., mug cup coffee)]
122. //VideoSegment[about(., cats)]
123. //VideoSegment[about(., pope john paul II second)]
124. //VideoSegment[about(., white house fountain)]
```

C.2 TRECVID 2004 queries

```
125. //VideoSegment[about(., street multiple pedestrians motion multiple
       vehicles motion somewhere)]
126. //VideoSegment[about(., buildings flood waters around)]
127. //VideoSegment[about(., people dogs walking together)]
128. //VideoSegment[about(., congressman henry hyde face whole part
       angle)]
129. //VideoSegment[about(., zooming capitol dome)]
130. //VideoSegment[about(., hockey rink least nets fully point view)]
131. //VideoSegment[about(., fingers striking keys keyboard least
       partially)]
132. //VideoSegment[about(., people moving stretcher)]
133. //VideoSegment[about(., saddam hussein)]
134. //VideoSegment[about(., boris yeltsin)]
135. //VideoSegment[about(., sam donaldson face whole part angle
       including both eyes people)]
136. //VideoSegment[about(., person hitting golf ball then hole)]
137. //VideoSegment[about(., benjamin netanyahu)]
138. //VideoSegment[about(., people going down steps stairs)]
139. //VideoSegment[about(., handheld weapon firing)]
140. //VideoSegment[about(., bicycles rolling)]
141. //VideoSegment[about(., umbrellas)]
142. //VideoSegment[about(., tennis player contacting ball tennis
       racket)]
143. //VideoSegment[about(., wheelchairs may motorized)]
144. //VideoSegment[about(., bill clinton speaking least part flag
       behind)]
145. //VideoSegment[about(., horses motion)]
146. //VideoSegment[about(., skiers skiing slalom course least gate
       pole)]
147. //VideoSegment[about(., buildings fire flames smoke)]
148. //VideoSegment[about(., signs banners carried people march
       protest)]
```

Appendix D

INEX Multimedia track NEXI queries

```
1. //destination[about(.//images//image,  buddha
         src:/images/BN417_16.jpg)]
      //*[(about(., asia) or about(., asian)) and (about(., buddha)
      or about(., buddhist))]
2. //destination[(about(., church) and about(., europe)) or
      about(.//images//image, +church +cathedral) or
      about(.//images//image, src:/images/BN6082_10.jpg)]
3. //destination[about(., "national park") and about(., africa) and
      about(.//health, - malaria)]
      //images//image[about(., "national park")
      or about(., src:/images/BN1038_32.jpg)]
4. //destination[about(.//general//introduction, best nightlife)]
      //images//image[about(., night city view
         src:/images/BN145_202.jpg src:/images/BN31_39.jpg)]
5. //destination[about(., asia) and
      about(.//point_of_interest, hindu culture)]
      //images//image[about(., thaipusam festival
         src:/images/BN234_604.jpg)]
6. //destination[about(., europe) and
      about(.//culture//history, king queen)]
      //images//image[about(., royal palace residence
         src:/images/BN7386_10.jpg)]
7. //destination[about(., gambling in casino)]
      //image[about(., "glamour and glitter" city life casino
         src:/images/BN4079_17.jpg src:/images/BN4879_11.jpg)]
8. //images[about(.,river danube)]
9. //image[about(.,nightlife pubs clubs parties dancing drinking)]
10. //destination[about(.,children activities) AND
      about(.,"beautiful beaches") AND
      about(.//image, src:/images/BN14115_5.jpg)]
11. //destination[about(.//image, fruit vegetables
         src:/images/BN2787_4.jpg)]
      //point_of_interest[about(., food fruit vegetable market)]
12. //destination[about(.//attractions//destination, "sea fishing") OR
```

```
        about(.//activity, "sea fishing")]//image[about(., coast)]
13. //destination[about(., city river)]//images//image[about(., river city)
        and about(., src:/images/wg-brisbane-400x300.gif)]
14. //images//image[about(.,beach tree)]
15. //destination[about(., mountain and water) and
        about(.//activities, holiday)]
        //images//image[about(.,mountain water bay sea ocean river)
        and about(., src:/images/BN59706.jpg)]
16. //destination[about(., university)]//map[about(., university)]
17. //destination[about(., old city) AND about(.//image, palace) AND
        about(.//image, church)]
18. //destination[about(.//map,"latin america")]
19. //destination[(about(.//history, persia) or
        about(.//image, persian empire)) and about(.//image, mosque)]
        //(point_of_interest|destination)[about(., ancient city)]
20. //destination[about(., "rock museum") and
        .//environment//longditude>-130 and .//environment//longditude<-65
        and about(.//event, "christmas official holiday")]
        //map[about(., station)]
21. //destination[about(., pacific island) AND
        about(.//image, island trees plants or rainforest)]
22. //destination[about(., sunset) AND (about(.//image, sunset sun) OR
        about(.//image, trees mountain water sunset glow) )]
23. //destination[about(., ski resort) AND
        about(.//image, ski resort snow mountain)]
24. //destination[about(., us city) AND
        about(.//image, street building people)]
25. //destination[about(.//image, ancient ships and vessels)]
        //(activities|attractions)
           [about(.,harbour visits to ancient ships vessel)]
```

Bibliography

[1] S. Abiteboul. Querying Semi-Structured Data. In *Proceedings of the 6th International Conference on Database Theory (ICDT)*, pages 1–17. Springer, January 1997.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley Professional, 1st edition, November 1994.

[3] M. Abolhassani, N. Fuhr, and S. Malik. HyREX at INEX 2003. In N. Fuhr, S. Malik, and M. Lalmas, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 27–32. ERCIM Workshop Proceedings, March 2004.

[4] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[5] G. Amati, C. Carpineto, and G. Romano. Merging XML Indices. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 253–260. Lecture Notes in Computer Science 3493, Springer, March 2005.

[6] S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, D. McBeath, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 Full-Text. Technical Report WD-xmlquery-full-text–20051103, W3C, November 2005.

[7] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *Proceedings of the 13th International World Wide Web Conference*, pages 583–594, May 2004.

[8] S. Amer-Yahia and P. Case. XQuery and XPath Full-Text Use Cases. Technical Report WD-xmlquery-full-text-use-cases-20030214, W3C, February 2003.

[9] S. Amer-Yahia and P. Case. XQuery 1.0 and XPath 2.0 Full-Text Use Cases. Technical Report WD-xmlquery-full-text-use-cases-20060501, W3C, May 2006.

[10] R. Baeza-Yates and G. Navarro. Integrating Contents and Structure in Text Retrieval. *SIGMOD Records*, 25(1):67–79, 1996.

[11] R. Baeza-Yates and G. Navarro. XQL and Proximal Nodes. *Journal of the American Society for Information Science and Technology (JASIST)*, 53(6):504–514, 2002.

[12] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley Longman, ACM Press edition, May 1999.

[13] E. M. Bakker, T. S. Huang, M. S. Lew, N. Sebe, and X. S. Zhou, editors. *Image and Video Retrieval, Second International Conference, CIVR*, volume 2728 of *Lecture Notes in Computer Science*. Springer, July 2003.

[14] A. Berglund, S. Boag, D. Chamberlin, M. Fernández, M. Kay, J. Robie, and J. Siméon. XML Path Language (XPath) 2.0. Technical Report WD-xpath20-20041029, W3C, October 2004.

[15] P. Biron and A. Malhotra. XML Schema Part 2: Datatypes Second Edition. Technical Report REC-xmlschema-2-20041028, W3C, October 2004.

[16] H. E. Blok. *Database Optimization Aspects for Information Retrieval.* PhD thesis, University of Twente, April 2002.

[17] H. E. Blok, V. Mihajlović, G. Ramírez, T. Westerveld, D. Hiemstra, and A. P. de Vries. The TIJAH XML Information Retrieval System. In S. Dumais, E. N. Efthimiadis, D. Hawking, and K. Järvelin, editors, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 725. ACM Press, August 2006.

[18] S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. Technical Report WD-xquery-20041029, W3C, October 2004.

[19] P. Boncz. *Monet: A Next Generation Database Kernel for Query Intensive Applications.* PhD thesis, CWI, May 2002.

[20] P. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teubner. MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine. In *Proceedings of the SIGMOD International Conference on Management of Data*, June 2006.

[21] B. Bos. The XML Data Model. Technical report, W3C, April 1997.

[22] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). Technical Report REC-xml-20040204, W3C, February 2004.

[23] J. Broglio, J. P. Callan, and W. B. Croft. INQUERY System Overview. In *Proceedings of the TIPSTER Text Program (Phase 1) Workshop*, pages 47–67. Morgan Kaufmann, 1994.

[24] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic Query Expansion Using SMART: TREC 3. In D. Harman, editor, *Proceedings of the 3rd Text Retrieval Conference (TREC)*, pages 69–80. National Institute of Standards and Technology (NIST), 1994.

[25] D. Bulterman, G. Grassel, J. Jansen, A. Koivisto, N. Layaïda, T. Michel, S. Mullender, and D. Zucker. Synchronized Multimedia Integration Language (SMIL 2.1). Technical Report REC-SMIL2-20051213, W3C, December 2005.

[26] P. Buneman. Semistructured Data. In *Proceedings of the 16th ACM SIGACT - SIGMOD - SIGART Symposium on Principles of Database Systems*, pages 117–121. ACM press, May 1997.

[27] F. Burkowski. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Data. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–125, 1992.

[28] S. Buxton and M. Rys. XQuery and XPath Full-Text Requirements. Technical Report WD-xquery-full-text-requirements-20030502, W3C, May 2003.

[29] J. P. Callan, W. B. Croft, and J. Broglio. TREC and TIPSTER Experiments with INQUERY. *Information Processing and Management: an International Journal*, 31(3):327–343, 1995.

[30] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications (DEXA)*, pages 78–83, 1992.

[31] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML Documents via XML Fragments. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158. ACM Press, July 2003.

[32] D. D. Chamberlin, M. M. Astrahan, M. W. Blasgen, J. N. Grey, W. F. King, B. G. Lindsey, R. Lorie, J. W. Mehl, T. G. Price, F. Putzolu, P. G. Selinger, M. Schkolnick, D. R. Slutz, I. L. Traiger, B. W. Wade, and R. A. Yost. A History and Evaluation of System R. *Communications of the ACM*, 24(10):377–387, 1981.

[33] S.-F. Chang, W. Hsu, L. Kennedy, L. Xie, A. Yanagawa, E. Zavesky, and D.-Q. Zhang. Columbia University TRECVID-2005 Video Search and High-Level Feature Extraction. In *Proceedings of the TRECVID Workshop*, 2005.

[34] S. Chaudhuri, R. Ramakrishnan, and G. Weikum. Integrating DB and IR Technologies: What is the Sound of One Hand Clapping? In M. Stonebraker, G. Weikum, and D. DeWitt, editors, *Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 1–12. ACM Press, January 2005.

[35] C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, and D. Wu. Structure and Performance of a Dependency Language Model. In *Proceedings of Eurospeech*, pages 2775–2778, 1997.

[36] M. G. Christel and A. G. Hauptmann. The Use and Utility of High-Level Semantic Features in Video Retrieval. In *Proceedings of the Conference on Image and Video Retrieval (CIVR)*, pages 134–144. Lecture Notes in Computer Science 3568, July 2005.

[37] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. Technical Report REC-xpath-19991116, W3C, November 1999.

[38] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski. An Algebra for Structured Text Search and a Framework for its Implementation. *The Computer Journal*, 38(1):43–56, 1995.

[39] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski. Schema-Independent Retrieval from Heterogeneous Structured Text. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 279–290, April 1995.

[40] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[41] M. Consens and T. Milo. Algebras for querying text regions: Expressive power and optimization. *Journal of Computer and System Sciences (JCSS)*, 57(3):272–288, December 1998.

[42] W. S. Cooper. Expected Search Length: A Single Measure of Retrieval Effectiveness Based on the Weak Ordering Action of Retrieval Systems. *American Documentation*, 19(1):30–41, 1968.

[43] W. S. Cooper. Getting Beyond Boole. *Information Processing and Management*, 24(3):243–248, 1988.

[44] G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and R. C. Good. The MultiText Retrieval System (demonstration abstract). In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 334. ACM Press, August 1999.

[45] G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and T. R. Lynam. MultiText Experiments for TREC. In E. M. Voorhees and D. K. Harman, editors, *TREC Experiments and Evaluation in Information Retrieval*, pages 347–372. MIT Press, 2005.

[46] J. Cowan and R. Tobin. XML Information Set (Second Edition). Technical Report REC-xml-infoset-20040204, W3C, February 2004.

[47] F. Crestani, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. A Multi-layered Bayesian Network Model for Structured Document Retrieval. In T. D. Nielsen and N. L. Zhang, editors, *Proceedings of the 7th European Conference Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 74–86. Lecture Notes in Computer Science, Springer, July 2003.

[48] W. B. Croft. Knowledge-Based and Statistical Approaches to Text Retrieval. *IEEE Expert: Intelligent Systems and Their Applications*, 8(2):8–12, 1993.

[49] W. B. Croft, R. Cook, and D. Wilder. Providing Government Information on the Internet: Experiences with THOMAS. In *Proceedings of the 2nd Annual Conference on the Theory and Practice of Digital Libraries (DL)*, pages 19–24. ERCIM Workshop Proceedings, June 1995.

[50] W. B. Croft and H. Turtle. A Retrieval Model for Incorporating Hypertext Links. In *Proceedings of the 2nd Annual ACM Conference on Hypertext (HYPERTEXT)*, pages 213–224. ACM Press, 1989.

[51] C. J. Crouch, A. Mahajan, and A. Bellamkonda. Flexible Retrieval Based on Vector Space Model. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 292–302. Lecture Notes in Computer Science 3493, Springer, March 2005.

[52] E. Curtmola, S. Amer-Yahia, P. Brown, and M. Fernàndez. GalaTex: A Conformant Implementation of the XQuery Full-Text Language. In A. Ellis and T. Hagino, editors, *Proceedings of the 14th International Conference on World Wide Web (WWW), Special interest tracks and posters*, pages 1024–1025. ACM Press, May 2005.

[53] P. Dadam, K. Kuespert, F. Andersen, H. Blanken, and R. Erbe. A DBMS Prototype to Support Extended NF2 Relations: An Integrated View on Flat Tables and Hierarchies. *SIGMOD Record*, 15(2):356–367, 1986.

[54] S. J. de Rose. *The SGML FAQ Book: Understanding the Foundation of HTML and XML*, volume 7 of *Electronic Publishing*. Kluwer Academic Publishers, July 1997.

[55] A. P. de Vries. Content Independence in Multimedia Databases. *Journal of the American Society for Information Science and Technology (JASIST)*, 52(11):954–960, 2001.

[56] A. P. de Vries and D. Hiemstra. The Mirror DBMS at TREC-8. In *Proceedings of the 8th Text Retrieval Conference (TREC)*, pages 725–734. National Institute of Standards and Technology (NIST), November 1999.

[57] A. P. de Vries, G. Kazai, and M. Lalmas. Tolerance to Irrelevance: A User-effort Oriented Evaluation of Retrieval Systems without Predefined Retrieval Unit. In *Proceedings of the Recherche d'Informations Assistee par Ordinateur (RIAO)*, pages 463–473, April 2004.

[58] A. P. de Vries, G. Kazai, and M. Lalmas. Evaluation Metrics 2004. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*. pre-proceedings, March 2005.

[59] A. P. de Vries, M. G. L. M. van Doorn, H. M. Blanken, and P. M. G. Apers. The miRRor MMDBMS Architecture. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, pages 758–761. Morgan Kaufmann Publishers Inc., September 1999.

[60] A. P. de Vries and A. Wilschut. On the Integration of IR and Databases. In *Proceedings of the 8th IFIP 2.6 Working Conference on Data Semantics*, pages 16–31, 1998.

[61] S. DeRose, R. Daniel, P. Grosso, E. Maler, J. Marsh, and N. Walsh. XML Pointer Langauge (XPointer). Technical Report WD-xptr-20020816, W3C, August 2002.

[62] S. DeRose, E. Maler, and D. Orchard. XML Linking Language (XLink) Version 1.0. Technical Report REC-xlink-20010627, W3C, June 2001.

[63] A. Doan and A. Halevy. Semantic Integration Research in the Database Community. *AI Magazine*, 26:83–94, 2005.

[64] E. N. Efthimiadis. Query Expansion. *Annual Review of Information Systems and Technology (ARIST)*, 31(4):121–187, 1996.

[65] M. Fernández, A. Malhotra, J. Marsh, M. Nagy, and N. Walsh. XQuery 1.0 and XPath 2.0 Data Model. Technical Report WD-xpath-datamodel-20041029, W3C, October 2004.

[66] D. Florescu, D. Kossmann, and I. Manolescu. Integrating Keyword Search into XML Query Processing. *Computer Networks*, 33(1-6):119–135, May 2000.

[67] N. Fuhr. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[68] N. Fuhr. A Probabilistic Relational Model for the Integration of IR and Databases. In R. Korfhage, E. M. Rasmussen, and P. Willett, editors, *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 309–317. ACM Press, June 1993.

[69] N. Fuhr. Models for Integrated Information Retrieval and Database Systems. *IEEE Data Engineering Bulletin*, 19(1):3–13, 1996.

[70] N. Fuhr. Models in Information Retrieval. In M. Agosti, F. Crestani, and G. Pasi, editors, *Lectures in Information Retrieval*, pages 21–50. Springer, 2000.

[71] N. Fuhr and K. Grossjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, September 2001.

[72] N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM Transactions on Information Systems (TOIS)*, 22(2):313–356, 2004.

[73] N. Fuhr, K. Großjohann, and S. Kriewel. A Query Language and User Interface for XML Information Retrieval. In H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors, *Intelligent Search on XML*, volume 2818 of *Lecture Notes in Computer Science*, pages 59–75. Springer-Verlag, August 2003.

[74] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors. *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval*, volume 3493. Springer-Verlag, May 2005.

[75] N. Fuhr and T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Transactions on Information Systems (TOIS)*, 15(1):32–66, 1997.

[76] J.-L. Gauvain, L. Lamel, and G. Adda. The LIMSI Broadcast News Transcription System. *Speech Communication*, 37(1-2):89–108, 2002.

[77] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 211–223. Lecture Notes in Computer Science 3493, Springer, March 2005.

[78] G. Gonnet and F. Tompa. Mind Your Grammar: A New Approach to Modelling Text. In P. M. Stocker, W. Kent, and P. Hammersley, editors, *Proceedings of the 13th International Conference on Very Large Databases (VLDB)*, pages 339–346, September 1987.

[79] N. Gövert, M. Abolhassani, N. Fuhr, and K. Grossjohann. Content-Oriented XML Retrieval with HyREX. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *Proceedings of the 1st Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 13–17. ERCIM Workshop Proceedings, March 2003.

[80] T. Grabs, K. Böhm, and H.-J. Schek. PowerDB-IR: Information retrieval on Top of a Database Cluster. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)*, pages 411–418. ACM Press, 2001.

[81] T. Grabs and H.-J. Schek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In *XML and Information Retrieval Workshop - 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 2002.

[82] T. Grabs and H.-J. Schek. ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *Proceedings of the 1st Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 35–40. ERCIM Workshop Proceedings, March 2003.

[83] D. A. Grossman and O. Frieder. *Information Retrieval: Algorithms and Heuristics.* The Information Retrieval. Springer, 2nd edition, December 2004.

[84] D. A. Grossman, O. Frieder, D. O. Holmes, and D. C. Roberts. Integrating Structured Data and Text: A Relational Approach. *Journal of the American Society of Information Science (JASIS)*, 48(2):122–132, 1997.

[85] D. A. Grossman, D. O. Holmes, and O. Frieder. A Parallel DBMS Approach to IR in TREC-3. In *Proceedings of the 3rd Text Retrieval Conference (TREC)*, pages 279–288. National Institute of Standards and Technology (NIST), 1994.

[86] T. Grust. Accelerating XPath Location Steps. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 109–120. ACM Press, June 2002.

[87] T. Grust, J. Teubner, and M. van Keulen. Accelerating XPath Evaluation in Any RDBMS. *ACM Transactions on Database Systems (TODS)*, 29(1):91–131, March 2004.

[88] T. Grust, M. van Keulen, and J. Teubner. Staircase Join: Teach A Relational DBMS To Watch its (Axis) Steps. In *Proceedings of the 29th International Conference on Very Large Databases (VLDB)*, pages 524–535, September 2003.

[89] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *Proceedings of the ACM SIMOD Conference*, pages 16–27, 2003.

[90] D. Harman, R. Baeza-Yates, E. Fox, and W. Lee. *Inverted Files*, pages 28–43. Prentice-Hall Inc., 1992.

[91] D. Hawking, N. Craswell, F. Crimmins, and T. Upstill. How Valuable is External Link Evidence when Searching Enterprise Webs? In *Proceedings of the 15th Australasian Database Conference (ADC)*, pages 77–84. Australian Computer Society, Inc., 2004.

[92] D. C. Hay. *Requirements Analysis: From Business Views to Architecture.* Prentice Hall, 1st edition, August 2002.

[93] M. Hearst. *User Interfaces and Visualization*, chapter 5. In Baeza-Yates and Ribeiro-Neto [12], ACM Press edition, May 1999.

[94] A. Henrich and G. Robbert. Combining Multimedia Retrieval and Text Retrieval to Search Structured Documents in Digital Libraries. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries.* ERCIM Workshop Proceedings, December 2000.

[95] D. Hiemstra. *Using Language Models for Information Retrieval.* PhD thesis, University of Twente, January 2001.

[96] D. Hiemstra. A Database Approach to Content-based XML Retrieval. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *Proceedings of the 1st Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 53–58. ERCIM Workshop Proceedings, March 2003.

[97] D. Hiemstra and V. Mihajlović. A Database Approach to Information Retrieval: The Remarkable Relationship Between Language Models and Region Models. Technical Report 05-35, Centre for Telematics and Information Technology, August 2005.

[98] D. Hiemstra and V. Mihajlović. The Simplest Evaluation Measures for XML Information Retrieval that Could Possibly Work. In A. Trotman, M. Lalmas, and N. Fuhr, editors, *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 6–13, July 2005.

[99] D. Hiemstra, H. Rode, R. van Os, and J. Flokstra. PF/Tijah: Text Search in an XML Database System. In M. Beigbeder, W. Buntine, and W. G. Yee, editors, *Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR)*, pages 12–17. ACM Press, August 2006.

[100] Y. E. Ioannidis. Query Optimization. *The Computer Science and Engineering Handbook*, 45:1038–1057, 1996.

[101] J. Jaakkola and P. Kilpeläinen. Using sgrep for Querying Structured Text Files. In *Proceedings of SGML Finland 1996*, October 1996.

[102] J. Jaakkola and P. Kilpeläinen. Nested Text-Region Algebra. Technical Report C-1999-2, Department of Computer Science, University of Helsinki, 1999.

[103] B. J. Jansen. The Effect of Query Complexity on Web Searching Results. *Information Research*, 6(1), 2000.

[104] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic Image Annotation and Retrieval using Cross-Media Relevance Models. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 2003.

[105] J. Jiang and C. Zhai. Extraction of Coxerent Relevant Passages using Hidden Markov Models. *ACM Transaction on Information Systems (TOIS)*, 24(3):295–319, 2006.

[106] M. Jiang, E. Jensen, S. Beitzel, and S. Argamon. Choosing the Right Bigrams for Information Retrieval. In *Proceeding of the Meeting of the International Federation of Classification Societies*, 2004.

[107] D. Jurafski and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice-Hall, 2000.

[108] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length Normalization in XML Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87. ACM Press, September 2004.

[109] J. Kamps, G. Mishne, and M. de Rijke. Language Models for Searching in Web Corpora. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the 13th Text Retrieval Conference (TREC)*, pages 250–261. National Institute of Standards and Technology (NIST), 2005.

[110] G. Kazai and M. Lalmas. INEX 2005 Evaluation Measures. In N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors, *Proceedings of the 4th Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 16–29. Lecture Notes in Computer Science 3977, Springer, March 2006.

[111] G. Kazai, M. Lalmas, and A. de Vries. The Overlap Problem in Content-oriented XML Retrieval Evaluation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79. ACM Press, July 2004.

[112] M. Kifer, A. Bernstein, and P. M. Lewis. *Database Systems: An Application-Oriented Approach.* Addison-Wesley, 2nd edition, 2006.

[113] W. Kraaij and R. Pohlmann. Porter's Stemming Algorithm for Dutch. In L. G. M. Noordman and W. A. M. de Vroomen, editors, *Informatiewetenschap 1994: Wetenschappenlijke bijdragen aan de derde STINFON Conferentie*, pages 167–180, 1994.

[114] W. Kraaij, A. F. Smeaton, and P. Over. TRECVID 2004 – An Overview. In *TREC Video Retrieval Evaluation Online Proceedings (TRECVID)*. http://www-nlpir.nist.gov/projects/tvpubs/tvpapers04/tv4overview.pdf, February 2004.

[115] J. Lafferty and C. Zhai. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119. ACM Press, 2001.

[116] M. Lalmas, T. Rölleke, and N. Fuhr. *Intellignet Retrieval of Hypermedia Documents*, volume 111, pages 325–347. Springer-Verlag, 2002.

[117] V. Lavrenko and W. B. Croft. Relevance-Based Language Model. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, 2001.

[118] C. Lee and G. Lee. Probabilistic Information Retrieval Model for a Dependency Structured Indexing System. *Information Processing and Management*, 41(2):161–175, March 2005.

[119] J. H. Lee. Analyzing the Effectiveness of Extended Boolean Models in Information Retrieval. Technical Report TR95-1501, Cornell University, 1995.

[120] A. Y. Levy, I. S. Mumick, and Y. Sagiv. Query Optimization by Predicate Move-Around. In *Proceedings of the 20th Very Large Database Conference (VLDB)*, pages 96–107, September 1994.

[121] J. List and A. P. de Vries. CWI at INEX 2002. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *Proceedings of the 1st Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 47–52. ERCIM Workshop Proceedings, March 2003.

[122] J. List, V. Mihajlović, A. P. de Vries, G. Ramírez, and D. Hiemstra. The TIJAH XML-IR System at INEX 2003. In *Proceedings of the 2nd Initiative on the Evaluation of XML Retrieval (INEX)*, pages 102–109. ERCIM Workshop Proceedings, March 2004.

[123] J. List, V. Mihajlović, G. Ramírez, A. P. de Vries, D. Hiemstra, and H. E. Blok. TIJAH: Embracing IR Methods in XML Databases. *Information Retrieval Journal*, 8(4):547–570, 2005.

[124] X. Liu and W. B. Croft. Passage Retrieval Based on Language Models. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 375–382. ACM Press, 2002.

[125] R. Luk, H. V. Leong, T. Dillon, A. Chan, W. B. Croft, and J. Allan. A Survey in Indexing and Searching XML Documents. *Journal of the American Society for Information Science and Technology (JASIST)*, 53(6):415–437, February 2002.

[126] M. E. Maron and J. L. Kuhns. On Relevance, Probabilistic Indexing and Information Retrieval. *Journal of ACM*, 7(3):216–244, 1960.

[127] J. Martinez. MPEG-7 Overview. Technical Report ISO/IEC JTC1/SC29/WG11, N6828, Internation Organisation for Standardization, October 2004.

[128] Y. Mass and M. Mandelbrod. Retrieving the most Relevant XML Components. In N. Fuhr, S. Malik, and M. Lalmas, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 12–18. ERCIM Workshop Proceedings, March 2004.

[129] Y. Mass and M. Mandelbrod. Component Ranking and Automatic Query Refinment for XML Retrieval. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 292–302. Lecture Notes in Computer Science 3493, Springer, March 2005.

[130] K. Masuda. A Ranking Model of Proximal and Structural Text Retrieval Based on Region Algebra. In K. Funakoshi, S. Kübler, and J. Otterbacher, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 50–57, 2003.

[131] K. Masuda, T. Ninomiya, Y. Miyao, T. Ohta, and J. Tsujii. A Robust Retrieval Engine for Proximal and Structural Search. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 58–60, 2003.

[132] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Technical Report REC-owl-features-20040210, W3C, February 2004.

[133] W. Meier. eXist: An Open Source Native XML Database. In A. B. Chaudri, M. Jeckle, E. Rahm, and R. Unland, editors, *Web-Services and Database Systems, Lecture Notes in Computer Science 2593*, pages 169–183. Springer, 2002.

[134] D. Metzler and W. B. Croft. Combining the Language Model and Inference Network Approaches to Retrieval. *Information Processing and Management: an International Journal*, 40(5):735–750, September 2004.

[135] D. Metzler, V. Lavrenko, and W. B. Croft. Formal Multiple-Bernoulli Models for Language Modeling. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 540–541, 2004.

[136] A. Micarelli, F. Gasparetti, F. Sciarrone, and S. Gauch. Personalized Search on the World Wide Web. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization.* Springer Verlag, 2006.

[137] V. Mihajlović. Score Region Algebra: A Framework for Structured IR. In G. Marchionini, A. Moffat, J. Tait, R. Baeza-Yates, and N. Ziviani, editors, *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (abstract)*, page 685. ACM Press, August 2005.

[138] V. Mihajlović, H. E. Blok, D. Hiemstra, and P. M. G. Apers. Score Region Algebra: Building a Transparend XML-IR Database. In A. Chowdhury, N. Fuhr, M. Ronthaler, H.-J. Schek, and W. Teiken, editors, *Proceedings of the 14th ACM International Conference on Information Knowledge and Management (CIKM)*, pages 12–19. ACM Press, March 2005.

[139] V. Mihajlović, D. Hiemstra, and H. E. Blok. Vague Element Selection and Query Rewriting for XML Retrieval. In F. de Jong and W. Kraaij, editors, *Proceedings of the 6th Dutch-Belgian Information Retrieval Workshop (DIR)*, pages 11–18. TNO ICT, March 2006.

[140] V. Mihajlović, D. Hiemstra, H. E. Blok, and P. M. G. Apers. An XML-IR-DB Sandwich: Is it Better with an Algebra in Between. In *Proceedings of the Joint Workshop on XML, IR and DB*, pages 31–38, July 2004.

[141] V. Mihajlović, D. Hiemstra, H. E. Blok, and P. M. G. Apers. Utilizing Structural Knowledge for Information Retrieval in XML Databases. Technical Report 05-19, Centre for Telematics and Information Technology, June 2005.

[142] V. Mihajlović, D. Hiemstra, H. E. Blok, and P. M. G. Apers. Exploiting Query Structure and Document Structure to Improve Document Retrieval Effectiveness. Technical Report 06-92, Centre for Telematics and Information Technology, October 2006.

[143] V. Mihajlović and M. Petković. Automatic Annotation of Formula 1 Races for Content-Based Video Retrieval. Technical Report 01-34, Centre for Telematics and Information Technology, December 2001.

[144] V. Mihajlović, G. Ramírez, A. P. de Vries, D. Hiemstra, and H. E. Blok. TIJAH at INEX 2004: Modeling Phrases and Relevance Feedback. In N. Fuhr,

M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 276–291. Lecture Notes in Computer Science 3493, Springer, March 2005.

[145] V. Mihajlović, G. Ramírez, T. Westerveld, D. Hiemstra, H. E. Blok, and A. P. de Vries. TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, and Overlap. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the 4th Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 72–87. Lecture Notes in Computer Science 3977, Springer, May 2006.

[146] D. Miller, T. Leek, and R. Schwartz. A Hidden Markov Model Information Retrieval System. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–221, 1999.

[147] G. Miller, C. Fellbaum, R. Tengi, S. Wolff, P. Wakefield, H. Langone, and B. Haskell. WordNet: A Lexical Database for the English Language. Technical report, Princeton University, 2005.

[148] R. C. Miller. *Lightweight Structure in Text*. PhD thesis, School of Computer Science, Carnegie Mellon University, May 2002.

[149] R. C. Miller and B. Myers. Lightweight Structured Text Processing. In *Proceedings of the 1999 USENIX Annual Technical Conference*, pages 131–144, June 1999.

[150] G. Navarro and R. Baeza-Yates. A Language for Queries on Structure and Contents of Textual Databases. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 93–101, 1995.

[151] G. Navarro and M. Ortega. IXPN: An Index-Based XPath Implementation. Technical Report TR/DCC-2003-5, Departamento de Ciencias de la Computación, July 2003.

[152] S.-Y. Neo, J. Zhao, M.-Y. Kan, and T.-S. Chua. Video Retrieval using High Level Features: Exploiting Query Matching and Confidence-based Weighting. In *Proceedings of the Conference on Image and Video Retrieval (CIVR)*, July 2006.

[153] K. Ng. A Maximum Likelihood Ratio Information Retrieval Model. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[154] P. Ogilvie and J. P. Callan. Language Models and Structured Document Retrieval. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *Proceedings of the 1st Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 33–41. ERCIM Workshop Proceedings, March 2003.

[155] P. Ogilvie and J. P. Callan. Using Language Models for Flat Text Queries in XML Retrieval. In N. Fuhr, S. Malik, and M. Lalmas, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 12–18. ERCIM Workshop Proceedings, March 2004.

[156] P. Ogilvie and J. P. Callan. Hierarchical Language Models for XML Component Retrieval. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 224–237. Lecture Notes in Computer Science 3493, Springer, March 2005.

[157] R. Ordelman. *Dutch Speech Recognition in Multimedia Information Retrieval.* PhD thesis, University of Twente, October 2003.

[158] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson. Terrier Information Retrieval Platform. In D. E. Losada and J. M. Fernàndez-Luna, editors, *Proceeding of the 27th European Conference on Information Retrieval Research (ECIR)*, pages 517–519. Lecture Notes in Computer Science, Springer, March 2005.

[159] C. D. Paice. The Automatic Generation of Literature Abstracts: An Approach Based on the Identification of Self-Indicating Phrases. In *Proceedings of the 3rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–191. Butterworth & Co., 1981.

[160] C. D. Paice. Soft Evaluation of Boolean Search Queries in Information Retrieval Systems. *Information Technology Research Development Applications*, 3(1):33–41, 1984.

[161] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., March 1988.

[162] J. Pehcevski and J. A. Thom. HiXEval: Highlighting XML Retrieval Evaluation. In N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors, *Proceedings of the 4th Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 43–57. Lecture Notes in Computer Science 3977, Springer, March 2006.

[163] J. Pehcevski, J. A. Thom, S. M. Tahaghoghi, and A.-M. Vercoustre. Hybrid XML Retrieval Revisited. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 153–167. Lecture Notes in Computer Science 3493, Springer, March 2005.

[164] J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. RMIT INEX Experiments: XML Retrieval Using Lucy/eXist. In N. Fuhr, S. Malik, and M. Lalmas,

editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 134–141. ERCIM Workshop Proceedings, March 2004.

[165] C. Peters, P. Clough, J. Gonzalo, G. J. F. Jones, M. Kluck, and B. Magnini, editors. *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum, (CLEF)*, volume 3491. Springer-Verlag, September 2005.

[166] M. Petković. *Content-Based Video Retrieval Supported by Database Technology.* PhD thesis, University of Twente, February 2003.

[167] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized Search. *Communications of the ACM*, 45(9):50–55, 2002.

[168] B. Piwowarski. EPRUM Metrics and INEX 2005. In N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors, *Proceedings of the 4th Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 30–42. Lecture Notes in Computer Science 3977, Springer, March 2006.

[169] B. Piwowarski and P. Gallinari. Expected Ratio of Relevant Units: A Measure for Structured Information Retrieval. In N. Fuhr, S. Malik, and M. Lalmas, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 158–166. ERCIM Workshop Proceedings, March 2004.

[170] B. Piwowarski, H.-T. Vu, and P. Gallinari. Bayesian Networks and INEX'03. In N. Fuhr, S. Malik, and M. Lalmas, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 12–18. ERCIM Workshop Proceedings, March 2004.

[171] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM Press, 1998.

[172] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[173] G. M. Quénot, D. Moraru, and L. Besacier. CLIPS at TRECVID: Shot Boundary Detection and Feature Detection. In *Proceedings of the TRECVID Workshop*, 2003.

[174] D. Raggett, A. L. Hors, and I. Jacobs. HTML 4.01 Specification. Technical Report REC-html401-19991224, W3C, December 1999.

[175] V. Raghavan, P. Bollmann, and G. S. Jung. A Critical Investigation on Recall and Precision as Measures of Retrieval System Performance. *ACM Transactions on Informatin Systems (TOIS)*, 7(3):205–229, 1989.

[176] E. Rahm and P. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal - The International Journal on Very Large Databases*, 10:334–350, 2001.

[177] G. Ramírez and A. P. de Vries. Combining Indexing Schemes to Accelerate Querying XML on Content and Structure. In V. Mihajlović and D. Hiemstra, editors, *Proceedings of the 1st Twente Data Management Workshop (TDM)*, pages 49–56. Centre for Telematics and Information Technology (CTIT), June 2004.

[178] G. Ramírez, T. Westerveld, and A. P. de Vries. Structural Features in Content Oriented XML Retrieval. Technical Report INS-E0508, Centre voor Wiskunde en Informatica (CWI), 2005.

[179] M. Rautiainen, M. Varanka, I. Hanski, M. Hosio, A. Parmila, J. Liu, T. Ojala, and T. Seppänen. TRECVID 2005 Experiments at MediaTeam Oulu. In *Proceedings of the TRECVID Workshop*, 2005.

[180] S. E. Robertson. Evaluation in Information Retrieval. In M. Agosti, F. Crestani, and G. Pasi, editors, *Proceedings of the 3rd European Summer School (ESSIR), Lectures on Information Retrieval*, pages 81–92. Springer-Verlag, September 2000.

[181] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241. Springer-Verlag, 1994.

[182] S. E. Robertson and S. Walker. Okapi at TREC-4. In D. Harman, editor, *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, pages 73–69, 1996.

[183] J. Robie, E. Derksen, P. Frankhauser, E. Howland, G. Huck, I. Macherius, M. Murata, M. Resnick, and H. Schöning. XQL (XML Query Language). Technical report, W3C, August 1999.

[184] T. Rölleke and N. Fuhr. *Information Retrieval with Probabilistic Datalog*, pages 221–243. Kluwer Academic Publishers, 1998.

[185] D. Rose and C. Stevens. V-Twin: A Lightweight Engine for Interactive Use. In *Proceedings of the 5th Text Retrieval Conference (TREC-5)*, pages 425–436, 1996.

[186] R. Rosenfeld. A Whole Sentence Maximum Entropy Language Model. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 230–237. IEEE Press, December 1997.

[187] R. Rosenfeld. Two Decades of Statistical Language Modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, August 2000.

[188] A. Salminen and F. Tompa. PAT Expressions: An Algebra for Text Search. In *Proceedings of the 2nd International Conference in Computational Lexicography (COMPLEX)*, pages 309–332, June 1992.

[189] G. Salton, J. Allan, and C. Buckley. Approaches to Passage Retrieval in Full Text Information Systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58. ACM Press, June 1993.

[190] G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[191] G. Salton and C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science and Technology (JASIST)*, 41(4):288–297, 1990.

[192] G. Salton, E. A. Fox, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[193] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* Computer Science. McGrow-Hill, Inc., 1st edition, 1983.

[194] K. Sauvagnat and M. Boughanem. Using Relevance Propagation Method for Adhoc and Heterogeneous Tracks at INEX 2004. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 337–248. Lecture Notes in Computer Science 3493, Springer, March 2005.

[195] P. Schäuble. SPIDER: A Multiuser Information Retrieval System for Semistructured and Dynamic Data. In R. Korfhage, E. M. Rasmussen, and P. Willett, editors, *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–327. ACM Press, June 1993.

[196] H.-J. Schek and P. Pistor. Data Structures for an Integrated Data Base Management and Information Retrieval System. In *Proceedings of the 8th International Conference on Very Large Data Bases (VLDB)*, pages 197–207, September 1982.

[197] F. Schiettecatte. Document Retrieval Using the MPS Information Server (a report on the trec-6 experiment). In E. Voorhees and D. Harman, editors, *Proceedings of the 6th Text Retrieval Conference TREC-6*, pages 477–488. National Institute of Standards and Technology (NIST), 1998.

[198] T. Schlieder and H. Meuss. Querying and Ranking XML Documents. *Journal of the American Society for Information Science and Technology (JASIST)*, 53(6):489–503, 2002.

[199] A. Schmidt, M. Kersten, M. Windhouwer, and F. Waas. Efficient Relational Storage and Retrieval of XML Documents. In *Proceedings of the 3rd International Workshop of the World Wide Web and Databases (WebDB)*, pages 47–52, May 2000.

[200] M. E. Senko. DIAM II: The Binary Infological Level and Its Database Language - FORAL. In *Proceedings of the Conference on Data: Abstraction, Definition and Structure*, pages 121–140. ACM Press, March 1976.

[201] U. Shah, T. Finin, and A. Joshi. Information Retrieval on the Semantic Web. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 461–468. ACM Press, 2002.

[202] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-based Approach to XML Retrieval. In N. Fuhr, S. Malik, and M. Lalmas, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 12–18. ERCIM Workshop Proceedings, March 2004.

[203] A. Skonnard and M. Gudgin. *Essential XML Quick Reference: A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More.* The DevelopMentor. Addison Wesley Professional, 1st edition, October 2001.

[204] A. F. Smeaton, W. Kraaij, and P. Over. TRECVID 2003 - An Overview. In *Proceedings of the TRECVID Workshop*, 2003.

[205] F. Song and W. B. Croft. A General Language Models for Information Retrieval. In *Proceedings of the 8th international Conference on Information and Knowledge Management (CIKM)*, pages 316–321, 1999.

[206] M. Srikanth and R. Srihari. Incorporating Query Term Dependencies in Language Models for Document Retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 405–406. ACM Press, 2003.

[207] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A Language Model Based Search Engine for Complex Queries. In *Proceedings of the International Conference on Intelligence Analysis*, May 2005.

[208] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–47. ACM Press, July 2003.

[209] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 16–40. Lecture Notes in Computer Science 3493, Springer, March 2005.

[210] D. Tsichritzis and A. C. Klug. The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems. *Information Systems*, 3(1):173–191, 1978.

[211] H. Turtle and W. B. Croft. Inference Networks for Document Retrieval. In J. Vidick, editor, *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24. ACM Press, 1990.

[212] M. van Keulen, J. Vonk, A. P. de Vries, J. Flokstra, and H. E. Blok. Moa and the Multi-model Architecture: A New Perspective on NF2. In V. V. Marik, W. Retschitzegger, and O. Stepankova, editors, *Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA)*, pages 67–76. Springer-Verlag, September 2003.

[213] C. J. van Rijsbergen. *Information Retrieval*. Department of Computer Science, University of Glasgow, 2nd edition, 1979.

[214] S. R. Vasanthakumar, J. P. Callan, and W. B. Croft. Integrating INQUERY with an RDBMS to Support Text Retrieval. *IEEE Data Engineering Bulletin*, 19(1), 1996.

[215] J.-N. Vittaut, B. Piwowarski, and P. Gallinari. An Algebra for Structured Queries in Bayesian Networks. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 292–302. Lecture Notes in Computer Science 3493, Springer, March 2005.

[216] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 1st edition, 2005.

[217] T. Westerveld. *Using Generative Probabilistic Models for Multimedia Retrieval*. PhD thesis, University of Twente, 2004.

[218] T. Westerveld, J. C. van Gemert, R. Cornacchia, D. Hiemstra, and A. P. de Vries. An Integrated Approach to Text and Image Retrieval: The Lowlands Team at Trecvid 2005. In *Proceedings of the TRECVID Workshop*, 2005.

[219] O. Wilde. *The Selfish Giant*. Putnam Juvenile, March 1995.

[220] M. Wu, P. Thomas, and D. Hawking. TREC 14 Enterprise Track at CSIRO and ANU. In *Proceedings of the 14th Text Retrieval Conference (TREC-14)*. National Institute of Standards and Technology (NIST), November 2005.

[221] J. Xu and W. B. Croft. Query Expansion Using Local and Global Document Analysis. In H.-P. Frei, D. Harman, P. Schaüble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11. ACM Press, 1996.

[222] M. Young-Lai and F. W. Tompa. One-Pass Evaluation of Region Algebra Expresions. *Information Systems*, 28(3):159–168, May 2003.

[223] C. Zhai and J. Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proceedings on the 24th Annual International ACM SIGIR Conferenceon Research and Development in Information Retrieval*, pages 334–342. ACM Press, 2001.

[224] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 425–436, 2001.

[225] X. Zhu and R. Rosenfeld. Improving Trigram Language Modeling With the World Wide Web. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 533–536, 2001.

[226] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted Files Versus Signature Files for Text Indexing. *ACM Transactions on Database Systems*, 23(4):453–490, 1998.

SIKS Dissertation Series

1998-01 Johan van den Akker (CWI), *DEGAS - An Active, Temporal Database of Autonomous Objects*

1998–02 Floris Wiesman (UM), *Information Retrieval by Graphically Browsing Meta-Information*

1998-03 Ans Steuten (TUD), *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*

1998-04 Dennis Breuker (UM), *Memory versus Search in Games*

1998-05 E.W.Oskamp (RUL), *Computerondersteuning bij Straftoemeting*

1999-01 Mark Sloof (VU), *Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products*

1999-02 Rob Potharst (EUR), *Classification using decision trees and neural nets*

1999-03 Don Beal (UM), *The Nature of Minimax Search*

1999-04 Jacques Penders (UM), *The practical Art of Moving Physical Objects*

1999-05 Aldo de Moor (KUB), *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*

1999-06 Niek J.E. Wijngaards (VU), *Re-design of compositional systems*

1999-07 David Spelt (UT), *Verification support for object database design*

1999-08 Jacques H.J. Lenting (UM), *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.*

2000-01 Frank Niessink (VU), *Perspectives on Improving Software Maintenance*

2000-02 Koen Holtman (TUE), *Prototyping of CMS Storage Management*

2000-03 Carolien M.T. Metselaar (UVA), *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.*

2000-04 Geert de Haan (VU), *ETAG, A Formal Model of Competence Knowledge for User Interface Design*

2000-05	Ruud van der Pol (UM), *Knowledge-based Query Formulation in Information Retrieval.*
2000-06	Rogier van Eijk (UU), *Programming Languages for Agent Communication*
2000-07	Niels Peek (UU), *Decision-theoretic Planning of Clinical Patient Management*
2000-08	Veerle Coup (EUR), *Sensitivity Analyis of Decision-Theoretic Networks*
2000-09	Florian Waas (CWI), *Principles of Probabilistic Query Optimization*
2000-10	Niels Nes (CWI), *Image Database Management System Design Considerations, Algorithms and Architecture*
2000-11	Jonas Karlsson (CWI), *Scalable Distributed Data Structures for Database Management*
2001-01	Silja Renooij (UU), *Qualitative Approaches to Quantifying Probabilistic Networks*
2001-02	Koen Hindriks (UU), *Agent Programming Languages: Programming with Mental Models*
2001-03	Maarten van Someren (UvA), *Learning as problem solving*
2001-04	Evgueni Smirnov (UM), *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
2001-05	Jacco van Ossenbruggen (VU), *Processing Structured Hypermedia: A Matter of Style*
2001-06	Martijn van Welie (VU), *Task-based User Interface Design*
2001-07	Bastiaan Schonhage (VU), *Diva: Architectural Perspectives on Information Visualization*
2001-08	Pascal van Eck (VU), *A Compositional Semantic Structure for Multi-Agent Systems Dynamics.*
2001-09	Pieter Jan 't Hoen (RUL), *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*
2001-10	Maarten Sierhuis (UvA), *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design*
2001-11	Tom M. van Engers (VUA), *Knowledge Management: The Role of Mental Models in Business Systems Design*
2002-01	Nico Lassing (VU), *Architecture-Level Modifiability Analysis*

2002-02	Roelof van Zwol (UT), *Modelling and searching web-based document collections*
2002-03	Henk Ernst Blok (UT), *Database Optimization Aspects for Information Retrieval*
2002-04	Juan Roberto Castelo Valdueza (UU), *The Discrete Acyclic Digraph Markov Model in Data Mining*
2002-05	Radu Serban (VU), *The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents*
2002-06	Laurens Mommers (UL), *Applied legal epistemology; Building a knowledge-based ontology of the legal domain*
2002-07	Peter Boncz (CWI), *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*
2002-08	Jaap Gordijn (VU), *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*
2002-09	Willem-Jan van den Heuvel (KUB), *Integrating Modern Business Applications with Objectified Legacy Systems*
2002-10	Brian Sheppard (UM), *Towards Perfect Play of Scrabble*
2002-11	Wouter C.A. Wijngaards (VU), *Agent Based Modelling of Dynamics: Biological and Organisational Applications*
2002-12	Albrecht Schmidt (Uva), *Processing XML in Database Systems*
2002-13	Hongjing Wu (TUE), *A Reference Architecture for Adaptive Hypermedia Applications*
2002-14	Wieke de Vries (UU), *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
2002-15	Rik Eshuis (UT), *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
2002-16	Pieter van Langen (VU), *The Anatomy of Design: Foundations, Models and Applications*
2002-17	Stefan Manegold (UVA), *Understanding, Modeling, and Improving Main-Memory Database Performance*
2003-01	Heiner Stuckenschmidt (VU), *Ontology-Based Information Sharing in Weakly Structured Environments*
2003-02	Jan Broersen (VU), *Modal Action Logics for Reasoning About Reactive Systems*
2003-03	Martijn Schuemie (TUD), *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*

2003-04	Milan Petković (UT), *Content-Based Video Retrieval Supported by Database Technology*
2003-05	Jos Lehmann (UVA), *Causation in Artificial Intelligence and Law - A modelling approach*
2003-06	Boris van Schooten (UT), *Development and specification of virtual environments*
2003-07	Machiel Jansen (UvA), *Formal Explorations of Knowledge Intensive Tasks*
2003-08	Yongping Ran (UM), *Repair Based Scheduling*
2003-09	Rens Kortmann (UM), *The resolution of visually guided behaviour*
2003-10	Andreas Lincke (UvT), *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*
2003-11	Simon Keizer (UT), *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
2003-12	Roeland Ordelman (UT), *Dutch speech recognition in multimedia information retrieval*
2003-13	Jeroen Donkers (UM), *Nosce Hostem - Searching with Opponent Models*
2003-14	Stijn Hoppenbrouwers (KUN), *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
2003-15	Mathijs de Weerdt (TUD), *Plan Merging in Multi-Agent Systems*
2003-16	Menzo Windhouwer (CWI), *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses*
2003-17	David Jansen (UT), *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
2003-18	Levente Kocsis (UM), *Learning Search Decisions*
2004-01	Virginia Dignum (UU), *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2004-02	Lai Xu (UvT), *Monitoring Multi-party Contracts for E-business*
2004-03	Perry Groot (VU), *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
2004-04	Chris van Aart (UVA), *Organizational Principles for Multi-Agent Architectures*
2004-05	Viara Popova (EUR), *Knowledge discovery and monotonicity*

2004-06	Bart-Jan Hommes (TUD), *The Evaluation of Business Process Modeling Techniques*
2004-07	Elise Boltjes (UM), *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*
2004-08	Joop Verbeek(UM), *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politile gegevensuitwisseling en digitale expertise*
2004-09	Martin Caminada (VU), *For the Sake of the Argument; explorations into argument-based reasoning*
2004-10	Suzanne Kabel (UVA), *Knowledge-rich indexing of learning-objects*
2004-11	Michel Klein (VU), *Change Management for Distributed Ontologies*
2004-12	The Duy Bui (UT), *Creating emotions and facial expressions for embodied agents*
2004-13	Wojciech Jamroga (UT), *Using Multiple Models of Reality: On Agents who Know how to Play*
2004-14	Paul Harrenstein (UU), *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
2004-15	Arno Knobbe (UU), *Multi-Relational Data Mining*
2004-16	Federico Divina (VU), *Hybrid Genetic Relational Search for Inductive Learning*
2004-17	Mark Winands (UM), *Informed Search in Complex Games*
2004-18	Vania Bessa Machado (UvA), *Supporting the Construction of Qualitative Knowledge Models*
2004-19	Thijs Westerveld (UT), *Using generative probabilistic models for multimedia retrieval*
2004-20	Madelon Evers (Nyenrode), *Learning from Design: facilitating multidisciplinary design teams*
2005-01	Floor Verdenius (UVA), *Methodological Aspects of Designing Induction-Based Applications*
2005-02	Erik van der Werf (UM), *AI techniques for the game of Go*
2005-03	Franc Grootjen (RUN), *A Pragmatic Approach to the Conceptualisation of Language*
2005-04	Nirvana Meratnia (UT), *Towards Database Support for Moving Object data*

2005-05	Gabriel Infante-Lopez (UVA), *Two-Level Probabilistic Grammars for Natural Language Parsing*
2005-06	Pieter Spronck (UM), *Adaptive Game AI*
2005-07	Flavius Frasincar (TUE), *Hypermedia Presentation Generation for Semantic Web Information Systems*
2005-08	Richard Vdovjak (TUE), *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
2005-09	Jeen Broekstra (VU), *Storage, Querying and Inferencing for Semantic Web Languages*
2005-10	Anders Bouwer (UVA), *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
2005-11	Elth Ogston (VU), *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*
2005-12	Csaba Boer (EUR), *Distributed Simulation in Industry*
2005-13	Fred Hamburg (UL), *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
2005-14	Borys Omelayenko (VU), *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*
2005-15	Tibor Bosse (VU), *Analysis of the Dynamics of Cognitive Processes*
2005-16	Joris Graaumans (UU), *Usability of XML Query Languages*
2005-17	Boris Shishkov (TUD), *Software Specification Based on Re-usable Business Components*
2005-18	Danielle Sent (UU), *Test-selection strategies for probabilistic networks*
2005-19	Michel van Dartel (UM), *Situated Representation*
2005-20	Cristina Coteanu (UL), *Cyber Consumer Law, State of the Art and Perspectives*
2005-21	Wijnand Derks (UT), *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*
2006-01	Samuil Angelov (TUE), *Foundations of B2B Electronic Contracting*
2006-02	Cristina Chisalita (VU), *Contextual issues in the design and use of information technology in organizations*
2006-03	Noor Christoph (UVA), *The role of metacognitive skills in learning to solve problems*

2006-04	Marta Sabou (VU), *Building Web Service Ontologies*
2006-05	Cees Pierik (UU), *Validation Techniques for Object-Oriented Proof Outlines*
2006-06	Ziv Baida (VU), *Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling*
2006-07	Marko Smiljanić (UT), *XML schema matching – balancing efficiency and effectiveness by means of clustering*
2006-08	Eelco Herder (UT), *Forward, Back and Home Again - Analyzing User Behavior on the Web*
2006-09	Mohamed Wahdan (UM), *Automatic Formulation of the Auditor's Opinion*
2006-10	Ronny Siebes (VU), *Semantic Routing in Peer-to-Peer Systems*
2006-11	Joeri van Ruth (UT), *Flattening Queries over Nested Data Types*
2006-12	Bert Bongers (VU), *Interactivation - Towards an e-cology of people, our technological environment, and the arts*
2006-13	Henk-Jan Lebbink (UU), *Dialogue and Decision Games for Information Exchanging Agents*
2006-14	Johan Hoorn (VU), *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
2006-15	Rainer Malik (UU), *CONAN: Text Mining in the Biomedical Domain*
2006-16	Carsten Riggelsen (UU), *Approximation Methods for Efficient Learning of Bayesian Networks*
2006-17	Stacey Nagata (UU), *User Assistance for Multitasking with Interruptions on a Mobile Device*
2006-18	Valentin Zhizhkun (UVA), *Graph transformation for Natural Language Processing*
2006-19	Birna van Riemsdijk (UU), *Cognitive Agent Programming: A Semantic Approach*
2006-20	Marina Velikova (UvT), *Monotone models for prediction in data mining*
2006-21	Bas van Gils (RUN), *Aptness on the Web*
2006-22	Paul de Vrieze (RUN), *Fundaments of Adaptive Personalisation*
2006-23	Ion Juvina (UU), *Development of Cognitive Model for Navigating on the Web*

2006-24	Laura Hollink (VU), *Semantic Annotation for Retrieval of Visual Resources*
2006-25	Madalina Drugan (UU), *Conditional log-likelihood MDL and Evolutionary MCMC*
2006-26	Vojkan Mihajlović (UT), *Score region algebra: a flexible framework for structured information retrieval*

Sažetak

Pre samo tri decenije istraživači su shvatili neophodnost struktuiranja podataka radi njihovog lakšeg skladištenja i radi lakšeg pristupa iz dana u dan sve većoj količini digitalnih podataka. Kao rezultat počelo se sa projektovanjem i razvijanjem sistema za upravljanje bazama podataka (engl. database management systems) čiji je cilj bio da čuvaju podatke na jednom mestu i da omoguće brzo pronalazenje značajnih informacija. Sa druge strane, velika količina tekstualnih podataka bila je čuvana i pristupalo joj se u nestrukturnom formatu. Pretraživanje takvih tekstualnih dokumenata radi pronalaženja informacija relevantnih za zadati upit otvoreno je istraživacko pitanje u oblasti koja nosi naziv pretraživanje informacija (engl. information retrieval). Istraživanja u toj oblasti dovela su do definisanja velikog broja modela za pretraživanje kao i do razvoja sistema za pretraživanje čiji je cilj rangiranje dokumenata prema procenjenom stepenu relevatnosti u odnosu na zadati upit. Iako su imali slične ciljeve, istraživačka oblast baza podataka i istraživačka oblast pretraživanja informacija razvijale su se uglavnom nezavisno jedna od druge. Nedavno, novi talas documenata 'zapretio' je da priblizi istraživanja u ove dve oblasti.

Ovaj talas je krenuo sa eksplozijom digitalnih documenata dostupnih preko Interneta. Za razliku od struktuiranja podataka u relacionim bazama podataka, cilj struktuiranja ovih dokumenata je vizuelna prezentacija i razmena podataka. Kao rezultat nastali su strukturni formati koji nemaju toliko stroge zahteve za struktuiranje podataka, nazvani polu-struktuirani formati (semi-structured formats). To su na primer *HyperText Markup Language* (HTML) i *eXtensible Markup Language* (XML). Kompaktno skladištenje ovako struktuiranih podataka, kao i brz i efikasan pristup informacijama koje se nalaze unutar njih, postali su novi problemi koji motivišu istraživačku zajednicu u poslednjih petnaestak godina. Pored toga, pretraživanje tako struktuiranih dokumenata ne mora da rezultira u ocenu koliko je neki dokument relevantan već i koliko su pojedini delovi (elementi unutar tog dokumenta) relevantni.

Oblast u koju se svrstava iztraživanje predstavljeno u ovoj tezi moze se ukratko opisati kao pronalaženje relevantnih informacija u polu-struktuiranim dokumentima. U okviru teze je opisan matematički aparat koji je korišćen za pretraživanje polu-struktuiranih dokumente koji sadrže logički i semantički opis strukture dokumenata, kao što su XML i SGML. Projektovanje i razvoj matematickog aparata za pretraživanje je u skladu sa preporukama koje dolaze kako iz prakse projektovanja i razvoja sistema za upravljanje bazama podataka, tako i sistema za pretraživanje informacija. U projektovanju je korišćena troslojna arhitektura sistema za upravljanje bazama podataka i korišćeni su modeli za pretraživanje razvijeni za sisteme za pretraživanje informacija.

Pre projektovanja i razvoja matematičkog aparata, problem pretraživanja struktuiranih dokumenata je analiziran i identifikovani su osnovni zahtevi koji se moraju poštovati prilikom razvoja sistema za pretraživanje. Ovi zahtevi su: (1) selekcija entiteta – selekcija različitih entiteta u okviru struktuiranih dokumenata, kao što su XML elementi, reči, atributi, lokacije slika i video materijala, a koji su deo upita; (2) odredjivanje stepena relevantnosti elemenata – odredjivanje stepena relevantnosti razlicitih elemenata uzimajući u obzir informacije koje sadrže; (3) kombinacija stepena relevantnosti – kombinacija stepena relevantnosti izmedju (različitih) elemenata unutar strukture koji rezultiraju u jedinstven stepen relevantnosti; (4) prenošenje stepena relevantnosti – prenošenje stepena relevantnosti od različitih elemenata do zajedničkog elementa pretka ili potomka u hierarhijskoj strukturi dokumenta. Ova četiri zahteva su osnova za razvoj algebre unutar baze podataka, koja je takodje u saglasnosti sa modelima za pretraživanje korišćenim za pretraživanje struktuiranih dokumenata. Prilikom definisanja algebre postojao je izazov koji je nazvani *transparentna implementacija modela za pretraživanje*, tj. mogućnost definisanja različitih modela za pretraživanje koji ne utiču na samu definiciju operatora u okviru algebre. Ovaj izazov je detaljnije opisan u nastavku u okviru tri aspekta.

Prvi aspekt se odnosi na definisanje algebre u kojoj je moguće specificirati modele za pretraživanje nezavisno od tipa korisničkog zahteva i bez uticaja na definiciju operatora u algebri, odnosno koja omogućuje *nezavisnost modela za pretraživanje*. Algebra je definisana polazeći od grupe algebri koje nose naziv prostorne algebre (engl. region algebras). Cilj prostornih algebri je da modeliraju pretraživanje struktuiranih dokumenata. Struktuirani dokumenti mogu se jednoznačno opisati koristeći intervale u prostoru tako što se svaki token (reč, oznaka, karakter) u okviru struktuiranih dokumenata može smatrati pozicijom u intervalu tekstualnog prostora. Ipak, postojeće algebre prostora ne sadrže operatore za odredjivanje stepena relevantnosti i za rangiranje. Iz tog razloga je izvršeno proširenje ovih algebri mehanizmom za odredjivanje stepena relevantnosti, tj. modelu podataka dodat je atribut stepena relevantnosti, a skup operatora proširen je operatorima za odredjivanje stepena relevantnosti. Nova algebra nosi naziv *prostorna algebra sa stepenima relevantnosti* (score region algebra – SRA). Operatori ove algebre su specificirani prateći dva cilja. Prvi je da operatori treba da budu u skladu sa četiri osnovna zahteva za razvoj sistema za pretraživanje struktuiranih dokumenata. Drugi je da operatori treba da obezbede nezavisnost modela za pretraživanje. Ova dva cilja dostignuta su zahvaljujući definisanju različitih operatora za svaki zahtev pri razvoju sistema za pretraživanje struktuiranih dokumenata, pri čemu se vrednost atributa stepena relevantnosti odredjuje korišćenjem apstraktnih funkcija i apstraktnih operatora. Tačna definicija apstraktnih funkcija i apstraktnih operatora odredjena je od strane administratora baze podataka (ili korisnika) i implementirana je na fizičkom nivou u okviru baze podataka.

Stepen uspešnosti je osnovna tema drugog aspekta. Cilj svakog sistema za pretraživanje je da dostigne što veći stepen uspešnosti na širokom spektru zadataka koji postoje u pretraživanju podataka. Kako bi ilustrovali stepen uspešnosti različitih modela za pretraživanje koji se mogu specificirati u okviru prostorne algebre sa stepenima relevantnosti, opsežno testiranje je predstavljeno u tezi. Testiranje je izvedeno korišćenjem XML i SGML kolekcija podataka na zadacima za pretraživanje delova dokumenata kao i za pretraživanje celih dokumenata. Prototip baze podataka pod nazivom “TIJAH”, čiji centralni deo čini prostorna algebra sa stepenima relevantnosti, korišćen je kao eksperimentalna platforma. Testiranja pokazuju da pri korišćenju različitih varijanti najpoznatijih modela za pretraživanje, “TIJAH” pokazuje zavidan stepen uspešnosti na oba zadataka. Ipak, najvažnije karakteristike prostorne algebre sa stepenima relevantnosti su da se može koristiti za analizu ponašanja različitih modela za pretraživanje koji su implementirani preko abstraktnih funkcija i abstraktnih operatora, kao i da se može upotrebiti za razvoj naprednijih modela za pretraživanje. U tezi je takodje pokazana korisnost algebre prilikom analize povezanosti izmedju implementacije različitih modela za pretraživanje sa jedne strane i različitih korisničkih upita i njihove kompleksnosti sa druge strane.

Treći aspect se odnosi na proširljivost i prilagodljivost sistema. Prostorna algebra sa stepenima relevantnosti (i “TIJAH” sistem) je prevashodno razvijena za pretraživanje u hijerarhijski struktuiranim dokumentima. Osim toga, algebra je razvijena da pored nezavisnosti modela za pretraživanje podrži i nezavisnost opisa sadržaja, tj. specifikaciju algebarskih operatora nezavisno od toga kako je sadržaj dokumenata modeliran na fizičkom nivou. Zahvaljujući tome prostorna algebra sa stepenima relevantnosti se može vrlo lako proširiti novim atributima u okviru modela podataka kao i novim operatorima. Novi operatori bi u tom slučaju sadržali nove funkcije za odredjivanje stepena relevantnosti, koje su zasnovane na dodatnom opisu sadržaja dokumenata. Tri primera objašnjena u tezi ilustruju ovakva proširenja. Prvi je proširenje modela podataka u okviru algebre dodatnim atributom koji opisuje dubinu ugnježdenja elemenata u strukturi. Drugi je specifikacija novih modela za pretraživanje, namenjenih pretraživanju video zapisa koji su modelirani kao tekstualni zapisi izgovorenih reči. Treći je uvodjenje novih operatora i definisanje novih modela za pretraživanje slika u kombinaciji sa tekstualnim pretraživanjem.

Samenvatting

Ongeveer dertig jaar geleden realiseerden onderzoekers zich dat gegevens gestructureerd moeten worden als men in staat zou willen zijn de grote stromen gegevens die elke dag geproduceerd worden te kunnen benaderen en opslaan. Een resultaat hiervan was het ontwerp en de ontwikkeling van *database management systemen* die gegevens centraal opslaan en die relevante gegevens terug kunnen vinden. Echter, een grote hoeveelheid tekstuele documenten werd nog steeds ongestructureerd opgeslagen. Het vinden van tekstuele documenten die relevante informatie bevatten op basis van een zoekvraag van de gebruiker is de afgelopen halve eeuw een belangrijke onderzoeksvraag geweest in het vakgebied van de *information retrieval*. *Information retrieval* onderzoek heeft tal van modellen en systemen opgeleverd, waarvan het doel het ordenen van de relevante documenten op basis van hun geschatte relevantie met betrekking tot een zoekvraag van de gebruiker is. Hoewel ze vergelijkbare doelen hebben, ontwikkelden de onderzoeksgebieden van *databases* en *information retrieval* zich vooral onafhankelijk van elkaar. Recentelijk 'dreigt' de nieuwe golf aan documenten deze vakgebieden dichter bij elkaar te brengen.

Deze golf begon met de nieuwe explosie van digitale documenten die nu publiek toegankelijk zijn op het Internet. Het structureren van deze documenten op een databasemanier, bijvoorbeeld in relationele tabellen, zou niet geschikt zijn voor presentatie en uitwisseling op het Web. Een resultaat hiervan was de ontwikkeling van gestructureerde formaten met minder randvoorwaarden aan de structuur, de zogenaamde semi-gestructureerde formaten. Dit zijn HyperText Markup Language (HTML) en eXtensible Markup Language (XML). Het efficiënt opslaan van zulke gestructureerde documenten, en het effectief en efficient zoeken naar informatie in de documenten, kwam in de laatste vijftien jaar op als een nieuw onderzoeksprobleem. Bovendien, het zoeken in gestructureerde documenten gaat niet alleen om het vinden van de meest relevante documenten, maar ook om het vinden van de meeste relevante componenten of elementen.

Het onderzoek dat in dit proefschrift wordt gepresenteerd richt zich op het zoeken naar relevante informatie in gestructureerde documenten. Het proefschrift beschrijft een raamwerk voor het zoeken van informatie (information retrieval) in documenten die annotaties gebruiken voor het beschrijven van hun logische en semantische structuur, zoals XML en SGML. De ontwikkeling van dit raamwerk voor het zoeken in gestructureerde informatie volgt de ideeën van de werelden van zowel *databases* als *information retrieval*. Het gebruikt een drie-lagen database architectuur en implementeert relevantieordening afgeleid van modellen voor *information retrieval*.

Voor het ontwikkelen van het raamwerk wordt het probleem van het zoeken in gestructureerde informatie geanalyseerd waarbij basiseisen voor gestructureerde zoeksystemen zijn vastgesteld. Deze eisen zijn: (1) selectie van entiteiten – het selecteren van verschillende entiteiten in gestructureerde documenten zoals elementen, termen, attributen, verwijzingen naar afbeeldingen en video, als onderdeel van de gebuikers zoekvraag; (2) relevantie score berekening van elementen – het berekenen van relevantiescores voor verschillende structuurelementen met betrekking tot de inhoud die ze bevatten; (3) combinatie van relevantiescores – het combineren van relevantiescores van (verschillende) elementen in een documentstructuur, resulterend in a gemeenschappelijke relevantiescore; (4) propagatie van relevantiescores – het propageren van relevantiescores van verschillende elementen naar gemeenschappelijke *ancestor* of *descendant* elementen volgens de zoekvraag. Aan deze vier eisen wordt voldaan door het ontwikkelen van een logische algebra in overeenstemming met de *information retrieval* modellen die gebruikt worden voor het ordenen van zoekresultaten. Bij het specificeren van de algebra gaan we de uitdaging van *transparante instantiatie van retrievalmodellen* aan, dat wil zeggen, de mogelijkheid om verschillende retrievalmodellen te definiëren zonder dat dat invloed heeft op de algebra operatoren. Deze uitdaging komt tot uiting in de drie onderstaande onderzoeksthema's.

Het eerste thema is de transparantie van het retrievalmodel; het ontwikkelen van een algebra waarin retrievalmodellen gespecificeerd kunnen worden, onafhankelijk van het type zoekvraag van de gebruiker en zonder de definitie van de algebra operatoren te beïnvloeden. De algebra welke wij ontwikkeld hebben is gebaseerd op regio algebra's. Het doel van regio algebra's is het modeleren van zoeken in gestructureerde documenten. Het toepassen van regio's op gestructureerde documenten is eenvoudig aangezien tokens in een gestructureerd document gezien kunnen worden als posities in een aaneengesloten tekst regio. Echter, geen van de traditionele regio algebra's kan gebruikt worden voor het orderen van documenten op relevantie. In ons onderzoek hebben wij deze algebra's daarom uitgebreid met een scorings mechanisme, dat wil zeggen, we hebben aan het datamodel een score attribuut toegevoegd en de set van operatoren uitgebreid met scorings mechanismen. Deze nieuw ontwikkelde algebra noemen wij *score region algebra* (SRA). SRA operatoren zijn gespecificeerd met twee doelen op het oog. Als eerste, het voldoen aan de vier basiseisen van *structured information retrieval*, en ten tweede, het mogelijk maken van transparantie van het retrievalmodel. Dit wordt bereikt door het definieren van verschillende operatoren voor iedere basiseis op een manier dat de waarde van het scorings attribuut bepaald wordt door abstracte functies en abstracte operatoren. De precieze definitie van deze abstracte functies en abstracte operatoren wordt bepaald door de database administrator (of gebruiker) en wordt ondersteund door een passende fysieke implementatie.

Het tweede onderzoeksthema heeft van doen met de effectiviteit van het zoeken. Ieder information retrieval systeem heeft tot doel om effectief te zijn op verscheidene zoektaken. Om de effectiviteit van verschillende retrievalmodellen geïnstantieerd in het SRA raamwerk te demonstreren is in dit proefschrift gebruik gemaakt van een grote hoeveelheid experimenten. Deze experimenten zijn gedaan door gebruik te maken van XML en SGML collecties op het gebied van *document retrieval* en *document component retrieval*. Voor de experimenten is een prototype van een database systeem gemaakt, genaamd TIJAH, gefocust op *score region algebra*'s. De experimenten laten zien dat, door gebruik te maken van varianten op state-of-the-art retrievalmodellen, ons systeem effectief is op zowel document retrieval als document component retrieval. Echter, de belangrijkste eigenschappen van het raamwerk zijn dat het gebruikt kan worden voor het analyseren van het gedrag van verscheidene retrievalmodellen geïmplementeerd langs de vier basiseisen van gestructureerde retrieval en dat het raamwerk gebruikt kan worden voor het ontwikkelen van effectievere retrievalmodellen. Ook laten we zien dat SRA nuttig is voor het bestuderen van relaties tussen aan de ene kant de implementatie van verschillende retrievalmodellen en aan de andere kant verschillende soorten zoekvragen van de gebruikers en de complexiteit hiervan.

Het derde en laatste onderzoeksthema gaat over de uitbreidbaarheid en flexibiliteit van het raamwerk. SRA (en het TIJAH systeem) is voornamelijk ontwikkeld voor het vinden van informatie in hiërarchisch gestructureerde documenten. Verder is het raamwerk, naast voor transparantie van het retrievalmodel, ook ontwikkeld voor het ondersteunen van transparantie van content beschrijving. Dit wil zeggen dat de verschillende operatoren van de algebra onafhankelijk zijn van de manier waarop content in het fysieke model wordt gemodelleerd. Als gevolg hiervan kan *score region algebra* eenvoudig worden uitgebreid met nieuwe attributen in het datamodel, evenals nieuwe operatoren. Deze nieuwe operatoren omvatten scorings functies welke de toegevoegde informatie over de inhoud van het document gebruiken. In dit proefschrift worden de uitbreidingen gedemonstreerd aan de hand van drie case studies. De eerste is een uitbreiding van het SRA datamodel met een attribuut voor het nesting-nivo van elementen, de twee is de introductie van een nieuw retrievalmodel voor het zoeken in spraak transcripties van video's en de derde is de introductie van nieuwe operatoren en nieuwe retrievalmodellen voor het zoeken in de combinatie van beeld en tekst.

Summary

Approximately three decades ago researchers realized that they would have to structure data to be able to store and access large amounts of data streams that were produced each day. As a result, database management systems were designed and developed, used to keep the data in one place and for finding relevant information in this data. On the other hand, a large amount of textual documents was still stored and accessed in unstructured format. Retrieval of such textual documents, containing relevant information with respect to a user query, has been an open research question studied in the information retrieval area for half a century. Information retrieval studies resulted in numerous retrieval models and retrieval systems whose goal is to rank relevant documents according to their estimated relevance to a user query. Although having similar goals research areas of databases and information retrieval developed mostly independently from each other. Recently, the new 'wave of documents' is 'threatening' to bring these two areas closer to each other.

This wave started with the new explosion of electronic documents, now publicly available on the Internet. Structuring these documents in a database way, for example in relational tables, would not be appropriate for their presentation and exchange over the Web. As a result structured formats with less structured constraints, called semi-structured formats, have been developed. These are HyperText Markup Language (HTML) and eXtensible Markup Language (XML). Efficient storage of such structured documents as well as effective and efficient retrieval of information in them emerged as a new research problem in the last fifteen years. Furthermore, retrieval in structured documents is not only about retrieving the most relevant documents but also about retrieving the most relevant document components (elements).

The scope of the research presented in this thesis is the retrieval of relevant information from structured documents. The thesis describes a framework for information retrieval in documents that have some form of annotation used for describing logical and semantical document structure, such as XML and SGML. The development of the structured information retrieval framework follows the ideas from both database and information retrieval worlds. It uses the three-level database architecture and implements relevance scoring mechanisms inherited from information retrieval models.

To develop the structured retrieval framework, the problem of structured information retrieval is analyzed and elementary requirements for structured retrieval systems are specified. These requirements are: (1) entity selection – the selection of different entities in structured documents, such as elements, terms, attributes, image and video references, which are parts of the user query; (2) entity relevance

score computation – the computation of relevance scores for different structured elements with respect to the content they contain; (3) relevance score combination – the combination of relevance scores from (different) elements in a document structure, resulting in a common element relevance score; (4) relevance score propagation – the propagation of scores from different elements to common ancestor or descendant elements following the query. These four requirements are supported when developing a database logical algebra in harmony with the retrieval models used for ranking. In the specification of the logical algebra we face a challenge of a *transparent instantiation of retrieval models*, i.e., the specification of different retrieval models without affecting the algebra operators. This challenge is portrayed in the three research issues discussed below.

The first issue is the development of an algebra where retrieval models can be specified in the algebra independently of the type of user requests and without affecting the definition of algebra operators, termed *retrieval model independence.* The algebra we developed is based on region algebras, whose aim is to model search in structured documents. The application of regions to structured documents is easy as tokens in a structured document can be seen as positions in a contiguous text region. However, none of the region algebras supports relevance ranking. We extended these algebras with a scoring mechanism, that is, we extended the data model with an extra score attribute, and extended the operator set with relevance scoring mechanisms. We call the new algebra *score region algebra* (*SRA*). SRA operators are specified with two goals in mind. First is to follow four elementary structured retrieval requirements. Second is to enable retrieval model independence. This is achieved through the definition of distinct operators for each structured retrieval requirement in such a way that the value of the scoring attribute is determined through abstract functions and abstract operators. The exact definition of these abstract functions and abstract operators is set by a database administrator (or user) and is supported by a proper physical implementation.

The second research issue is concerned with the retrieval effectiveness. The goal of every information retrieval system is to be effective on various information retrieval tasks. To demonstrate the effectiveness of various retrieval models instantiated within the SRA framework, extensive experimentation is presented in this thesis. The experiments are performed using XML and SGML collections on the tasks of document component retrieval and document retrieval. For the experimentation a prototype database system is developed, called TIJAH, centered around SRA. Experiments show that, using variants of state-of-the-art retrieval models, our system is quite effective on both tasks. However, the most important properties of the framework are that it can be used for analyzing the behavior of various retrieval models implemented along four structured retireval requirements and that it can be employed for developing more effective retrieval models. We also demonstrate the usefulness of SRA in studying relations between the implementation of different retrieval models on one side and different user requests and different query complexities on the other.

The third research issue is about the extensibility and flexibility of the framework. Score region algebra (and the TIJAH system) is predominantly developed for retrieval in hierarchically structured documents. Furthermore, besides retrieval model independence, the framework is developed to support content description independence. That is the independence of the specification of operators within the algebra from the way how content information is modeled at the physical level. As a consequence score region algebra can be easily extended with new attributes in the data model as well as with new operators. These new operators encompass relevance scoring functions that use additional document content information. In this thesis the extensions are demonstrated on three case studies. The first one is the extension of the SRA data model with the element nesting level attribute, the second one is the introduction of new retrieval models for retrieval from speech transcripts of a video, and the third one is the introduction of new operators and new retrieval models for image search in combination with text search.

Curriculum Vitae

Vojkan Mihajlović was born on 27 January 1978 in Niš, Serbia (former Yugoslavia). In 1992 he finished primary school "Ratko Vukićević" in Niš. He graduated from the "Bora Stanković" Gymnasium (Niš) in 1996. After graduation, he commenced his studies at the Faculty of Electrical Engineering, University of Niš. In 2002, he received the title of dipl. ing. in Computer Science, after defending his thesis "Detection of Highlights in Formula 1 Videos". The thesis was a result of a six months research performed at the University of Twente, Enschede, The Netherlands. From February 2002 until November 2002, he worked as a research assistant in CG & GISLAB at Faculty of Electrical Engineering in Niš, where he was involved in geographic information system (GIS) projects and in teaching courses.

From November 2002 until November 2006, he worked as a researcher in Database Group at the University of Twente in Enschede, The Netherlands. His research was within the Complex Information Retrieval Queries in a Database (CIRQUID) project. He worked on developing logical algebra for expressing ranked retrieval in structured documents. His research resulted in a number of publications in international journals, conferences, and workshops. Vojkan was also involved in organizing a workshop on XML and Information Retrieval and he was a reviewer for international journals and international conferences in the area of information retrieval. As of November 2006, he is working as a senior researcher on multimedia content analysis and retrieval within Storage Systems and Applications group at Philips Research in Eindhoven, The Netherlands.