From Document to Entity Retrieval

Improving Precision and Performance of Focused Text Search

Henning Rode

Samenstelling van de promotiecommissie:

Prof. dr. P.M.G. Apers	promotor
dr. ir. D. Hiemstra	assistent promotor
Prof. dr. ir. A.J. Mouthaan	voorzitter en secretaris
Prof. dr. W. Jonker	Universiteit Twente Philips Research, Eindhoven
Prof. dr. T.W.C. Huibers	Universiteit Twente Thaesis, Ede
Prof. dr. R. Baeza-Yates	Universidad de Chile, Santiago Universitat Pompeu Fabra, Barcelona Yahoo! Research, Barcelona
Prof. dr. M. Lalmas	Queen Mary University, London
dr. ir. A.P. de Vries	Technische Universiteit Delft Centrum Wiskunde & Informatica, Amsterdam

CTIT Ph.D. thesis Series No. 08-120
Centre for Telematics and Information Technology (CTIT)
P.O. Box 217 - 7500 AE Enschede - The Netherlands

SIKS Dissertation Series No. 2008-19
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover picture Eiko Braatz

ISBN 978-90-365-2689-0

ISSN 1381-3617 (CTIT Ph.D. thesis Series No. 08-120)

Printed by PrintPartners Ipskamp, Enschede, The Netherlands

FROM DOCUMENT TO ENTITY RETRIEVAL

IMPROVING PRECISION AND PERFORMANCE OF FOCUSED TEXT SEARCH

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Twente, op gezag van
de rector magnificus prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen op
vrijdag 27 juni 2008 om 15.00 uur

door

Henning Rode

geboren op 5 maart 1975
te Hannover (Duitsland)

Dit proefschrift is goedgekeurd door:

Prof. dr. P.M.G. Apers (promotor)
dr. ir. D. Hiemstra (assistent promotor)

to my grandfather
who should have gotten a PhD long before me

Acknowledgments

Writing a thesis that sums up my scientific work of four years was a new experience for me. First of all it asked quite some patience from myself. Instead of looking forward to new scientific challenges, it forced me to re-read, re-think, and re-write what I had done before. The confrontation with the past brought up old ideas, scientific plans, things I did as well as things I never found the time to do. And, last but not least, it made me think of all the people that accompanied me through that period and made it an exciting, enjoyable time.

First, I'd like to thank my supervisor Djoerd for all his detailed reviewing work on this thesis and on my other scientific writing, which improved the presentation "by far". But also for the nice working atmosphere we had during the whole period of my PhD, and for just being around for all kinds of questions and discussions starting on work issues but not always ending there.

There have been many more people though who contributed to this research work. My promoter Peter, who always tried to keep me on track, and without him I would probably not have finished my PhD in time. Furthermore, Pavel, Hugo, Claudia, Dolf, and Franciska, with whom I wrote papers together in this period, as well as Arjen, Vojkan, Arthur, Robin, and Mounia, who did an excellent job in reviewing my scientific work. All those people gave many fruitful input to my own work, and at the same time teached me to defend my own writing.

I also want to thank the database group at the UT for the good working environment and the friendly atmosphere; our soup cooperation for providing at least the remembrance of a warm lunch. To pick out a few people: It was Maurice who had the brilliant idea to ask me whether I would like to come to the Netherlands at a time when I was not really thinking of doing a PhD. Developing our own search system PF/Tijah would not have been that

successful and fun without our scientific programmer Jan, who helped me a lot with my code work when he was not climbing mountains at the remotest places of the world. Further, Sandra, Ida, and Suse could hardly have done more to support me or even shielding me from all kinds of administrative work and encouraged me in my first attempts of speaking Dutch. Finally, I want to mention my two office mates Vojkan and Arthur. We have not only shown to be a great office team, but also demonstrated how to survive nights at lonely island airports.

Science can be a tedious office job, but also a lot of fun, which I experienced early at our memorable farmhouse meetings, which turned normal scientist over night into cow traders and guitar heros. Thanks Thijs, Nina, Vojkan, Arjen, and Djoerd for these lively meetings and the motivation coming out of the discussions there.

Many people go to Barcelona for holidays. I went there for work, more precisely for an internship at Yahoo! Research, but made the strange experience that hard work and holiday feeling is not necessarily a contradiction. I'd specially like to thank all first-hour citizens of the research lab – Hugo, Massi, Jordi, Flavio, and Ricardo – for the inspiring work we did together and the nice summer in Barcelona I shared with all of you .

Fortunately, my PhD life had far more to offer than only a good scientific surrounding. I used to live together with quite a few rather different people whom I'm thankful for interrupting my scientific thoughts every evening: First, Lennard and Hendrik Jan, for all discussions about Dutch politics and especially Lennard for being such a strict Dutch teacher. Second, Woongroep 't Piepke – Sylvia, Frank, Marga, Martine, Robin, Marcel, and Jasper – for turning the slightly unimpressive Enschede into a place I really felt at home.

Though I was living and working abroad, the near-border position helped to keep close contact with many good friends in Germany, while at the same time making new friends in Enschede. I'd like to thank Markus, Andi, Eiko, Malve, Johanna, Caro, Basti, Wolfgang, Caro, Kerstin, Ursula, Mathias, Sveta, Vojkan, Sofka, Marko, and Tanya for the many good talks, about live, politics, religion and music, and for always treating me like I never moved so far away.

From the many music and sport activities I joined during my PhD time, I will pick out only one group here. "We just meet once a week and play some music together" was Dennis saying when he asked me to join the Gonnagles. It's about the best understatement he could have given for the most creative, enthousiastic, and lifely group of people I was ever part of. Thanks to Moes, Edwin, Marlies, Dennis, Daphne, Frank, Erik, Fayke, Marijn, Jaap, and Gijs for the special experience of being a Gonnagle.

Finally, I'm blessed to have a really great family, and with family I mean

all those people that gather in Benthe around christmas time. They supported me in whatever I was doing, inspired my scientific reasoning and contradiction, and they are probably one of the few families with whom you can sing songs in four voices. Special thanks to my parents Hanne and Rüdiger, my brother Holger, and all my grandparents Brunhild, Hermann, Lore, and Johannes who endured to spend so many years with me and had the biggest impact in making out of me the person I am now.

And last but not least, Carla who luckily never gave up spotting free places in my agenda to spend wonderful days, weekends and holidays together, and sharing with me all daily ups and downs in the nightly skype universe.

Henning Rode
Amsterdam, June 2008

x

Contents

1 Introduction **1**
- 1.1 From Document to Entity Retrieval . . . 2
- 1.2 Adaptivity in Text Search . . . 8
- 1.3 Research Objectives . . . 10

2 Document Retrieval **13**
- 2.1 Context Modeling for Information Retrieval . . . 14
  - 2.1.1 Conceptual Language Models . . . 17
- 2.2 Ranking Query and Meta-query . . . 18
  - 2.2.1 Combined Ranking of Query and Meta-Query . . . 19
- 2.3 Experiments . . . 21
- 2.4 Interactive Retrieval . . . 25
  - 2.4.1 Related Approaches . . . 26
- 2.5 Query-Profiles . . . 27
  - 2.5.1 Generating Temporal Profiles . . . 27
  - 2.5.2 Generating Topical Profiles . . . 29
  - 2.5.3 The Clarification Interface . . . 31
  - 2.5.4 Score Combination and Normalization . . . 33
- 2.6 Experiments . . . 34
- 2.7 Summary and Conclusions . . . 37

3 Structured Retrieval on XML **39**
- 3.1 Query Languages for Structured Retrieval . . . 39
  - 3.1.1 Structural Features of XML . . . 40
  - 3.1.2 General Query Language Requirements . . . 41
  - 3.1.3 NEXI . . . 42
  - 3.1.4 XQuery Full Text . . . 43

3.1.5 NEXI Embedding in XQuery . . . 45
3.2 Indexing XML Structure and Content . . . 47
3.2.1 Data Access Patterns . . . 47
3.2.2 Indices for Content and/or Structure . . . 48
3.2.3 The PF/Tijah Index . . . 51
3.2.4 Experiments . . . 54
3.3 Scoring XML Elements . . . 56
3.3.1 Containment Joins . . . 57
3.3.2 Experiments . . . 60
3.3.3 Query Plans . . . 62
3.3.4 Experiments . . . 65
3.4 Complex Queries . . . 68
3.4.1 Experiments . . . 72
3.5 Summary and Conclusions . . . 76

4 Entity Retrieval 79
4.1 Entity Retrieval Tasks . . . 80
4.2 Ranking Approaches for Entities . . . 83
4.3 Entity Containment Graphs . . . 87
4.3.1 Modeling Options . . . 88
4.4 Relevance Propagation . . . 91
4.4.1 One-Step Propagation . . . 93
4.4.2 Multi-Step Propagation . . . 94
4.5 Experimental Study I: Expert Finding . . . 97
4.5.1 Result Discussion . . . 99
4.6 Experimental Study II: Entity Ranking on Wikipedia . . . 106
4.6.1 Exploiting Document Entity Relations in Wikipedia . . . 106
4.6.2 Result Discussion . . . 108
4.7 Searching Mixed-typed Entities . . . 111
4.7.1 Model Adaptations . . . 111
4.7.2 Experiments . . . 113
4.8 Summary and Conclusions . . . 116

5 Review and Outlook 119
5.1 Review . . . 119
5.2 Outlook . . . 124

Bibliography 127

Summary 141

Samenvatting **143**

SIKS Dissertation Series **145**

1

Introduction

The vast availability of online information sources has essentially changed the way users search for information. We like to point out 3 main changes:

(1) Information retrieval has become a ubiquitous requirement for modern life. Looking for public transport connections, cultural activities, or searching for reviews on goods we want to buy are just examples of such often occurring search tasks in daily life. In contrast to the conventional scenario of information retrieval, where a person is spending hours in a library to find all information on a certain topic, we are often satisfied with just some useful information, but it needs to be found immediately.

(2) In the same way, people often do not look anymore for entire books or articles but for some specific information contained inside. Sometimes the wanted information is captured in one single document, but the user would need to find the right place; sometimes the necessary information is even spread over several documents. In both cases, a user would appreciate retrieval systems that arrange just the required bits of information appropriately.

(3) Users want to search different types of documents. Apart from the conventional sources of information, like books and articles, we also want to search nowadays in webpages, emails, blogs, or simply within a computer's file system.

The changes on search behavior ask among others for research in the following fields of information retrieval:

Performance Retrieval systems need to be able to come up with answers within seconds – better even within fractions of seconds – independent of

the size of the collection. With text collections growing faster than hardware performance is improving, this becomes a challenge for indices and scoring algorithms. We will use the term performance here only with respect to the execution time of a query, not – as often done otherwise – with respect to the quality of retrieval.

Precision With growing text resources, precision becomes more important than recall. Whereas still a large set of documents might contain a certain query term, we are in general only interested in – or satisfied with – a tiny subset. However, this subset has to contain the relevant information. Studies on the search behavior of users show, that if relevant documents are not found on top of the list, it is more likely that a user reformulates the query than that she/he looks for relevant documents further down the retrieved ranked list (Markey, 2007). Therefore, retrieval systems should provide a query language that gives means to specify precise queries and furthermore support the user reformulating the query. As a second consequence of the preference of precision over recall, the evaluation of retrieval systems needs to stress the importance of precision measures.

Structure Retrieval systems need to be aware of the structure of documents. When collections consist of heterogeneous types of documents, and/or the documents themselves are structured – for instance distinguishing by mark-up between representation code and content as in web pages – the indices of retrieval systems need to capture structural information of documents as well. We can also think of the aim to weigh query matches in the title or abstract of a document higher than in other parts. Furthermore, when users want to search explicitly for relevant parts within large documents, not only the index but also the query language needs to be able to express structural requirements.

This thesis combines research work that addresses the problems mentioned in the last three paragraphs. Improving precision as well as structural retrieval will be discussed together with performance issues of the proposed techniques.

1.1 From Document to Entity Retrieval

The user's interest in highly focused retrieval results is a common assumption in information retrieval. Instead of always getting entire documents, users want to see directly the relevant parts of long articles. In compliance with

this assumption, we will follow in this work a line from document, over XML, towards entity retrieval. It is also a progression from retrieval as we know it from the conventional library setting towards very focused retrieval of the smallest meaningful units in the text.

In fact, user behavior studies are not that clear about the above made base assumption (Malik et al., 2006; Larsen et al., 2006). When users were asked to choose appropriate entry points for reading a retrieved part of a longer text, they usually like to start at document level and not directly at the best ranked paragraph or sentence. This observation, which looks at first contradicting to the focused retrieval assumption, is in fact based on the users' experience with information retrieval systems returning irrelevant, inappropriate answers as well. We all are trained by the common web search engines to always check in the first place whether a given answer is indeed matching our information need and a trustable source of information. Apparently, such a check is easier when we are confronted with an entire web-page or document than with the best matching paragraph- or sentence-level retrieval results. This does not mean, however, that people are really interested in reading the whole article. A good indication for that is, that users often like keyword highlighting in the returned articles. Focused retrieval techniques are appreciated, but need to be accompanied by other views of the entire document to give evidence of the appropriateness of the found information. The problem will be discussed in more detail at other places of this thesis, but the task of finding suitable user interface designs will be left for research in the area of human computer interaction.

On the background of such user studies, the title of this thesis should not be misunderstood as a mission to "move" away from document retrieval. It is not claiming an evolutionary development from document to entity retrieval, but for diversification of retrieval techniques. Document retrieval will remain as important as it always was, but apart from that, we need more focused retrieval methods. In the same way, the chapters of this book do not outdate each other, but discuss methods for high precision retrieval on all such levels of text retrieval.

The call for focused retrieval techniques is not new, however. We will shortly summarize and compare the main retrieval characteristics on the different granularity levels of returned text units.

Document Retrieval Document retrieval regards each document as an atomic unit of interest. It is not distinguished whether parts of a document are relevant to an information need but others not. Looking at Figure 1.1, the user of a document retrieval system will find a link to the entire outlined

document if it was considered as relevant to her/his query. Also the relevance estimation is based on the content of the entire document. If one chapter of the visualized thesis is highly relevant, but the other chapters are not, the final relevance estimation of the entire document is considerably lower than those of short documents being exclusively about the topic of interest. Single documents are either one-to-one identical with single files, or special pre-defined (SGML or XML) markup is used, to determine the bounds of single documents within large collection files. From an indexing perspective, document retrieval allows the construction of efficient inverted document index structures. Neglecting special requests like the search for phrases, most document retrieval models think of a document as a *bag of words*. It is then not necessary to store the exact position of keywords within a document.

Passage Retrieval One of the early approaches towards more focused retrieval results was the so-called *passage retrieval*. *"When the stored document texts are long, the retrieval of complete documents may not be in the users' best interest"* (Salton et al., 1993). Passage retrieval leaves it open to the retrieval system to define the boundaries of an appropriate passage. In fact, finding the right cut-out of a text is seen as the major challenge of the approach. A passage retrieval system typically does not take into account the structure of a document as shown in Figure 1.1, but returns arbitrary text fragments. Typically text windows of a fixed number of words around the found keyword mentions are returned. Retrieval models are still applied on document level to achieve a ranked document list in a first step. Only thereafter documents are analyzed in order to return the most suitable passage according to the query. The spreading of matching keyword positions inside a document is taken into account here combined with sentence and paragraph recognition to return useful units of text. Compared to document retrieval, the

```
<document>
<title>From Document to Entity Retrieval</title>
<author> Henning Rode </author>
<date> 27th June 2008 </date>
<content>
<introduction>
The vast availability of online information sources has essentially changed the way users search for information. We want to point out 3 main changes:
⋮
<section no="1.1">
In fact, user behavior studies are not that clear about the above made base assumption ( Malik et al., 2006 ; Larsen et al., 2006 ).
⋮
that can be displayed in response to the selection of an entity.
</conclusions>
</content>
</document>
```

Figure 1.1: Elements of a Document

index of a passage retrieval system also needs to maintain word positions inside documents, which typically doubles the size of the term posting lists. Moreover, one should notice that the evaluation of passage retrieval systems becomes more complicated. Apart from the fuzziness of relevance itself also the boundaries of an appropriate text cut-out become a matter of subjective preferences.

Fielded Search Often documents come with markup (e.g. HTML, XML, or LaTeX), describing their text structure in a machine readable form. Assuming a homogeneous text collection, we might know in advance, which tagged *fields* contain information a user will search. Fielded retrieval allows then to constrain a query to a specific part of the text (e.g. title search) or to exclude non-textual fields like visualization code of HTML-pages. In the example document (Figure 1.1) the fields `title`, `author`, but also `section` could be used to narrow down the search space. Some systems are also able to combine scores of multiple fields to one final document score. In contrast to passage retrieval, the different fields are usually treated as "mini documents" for the applied retrieval models. Thus, statistics like document sizes, or term likelihoods are calculated according to the fields itself rather than the entire documents. On the other hand, it is typically not the aim to retrieve the text of the fields only, but still entire documents scored by their contained fields. The approach is consequently called "fielded search" and not "field retrieval". Early experiments in this area have been done by Wilkinson (1994) showing how weighted fielded search can improve standard document retrieval. Robertson et al. (2004) examined how common retrieval models fit to fielded search and how the models should be adapted for this purpose. Finally, there are many application areas for fielded search systems, first of all in so-called "known item search", where it is assumed that the user is able to clearly constrain the search space (Dumais et al., 2003).

Also the index of such systems usually maintains fields in the same way as documents. Hence, indexed fields have to be predefined by the user at indexing time already. Compared to passage retrieval mentioned before, fielded search is not trying to find the best text cut-out itself – the fields of interest are explicitly stated in the user query.

XML Retrieval Sometimes systems that enable fielded search are regarded as XML retrieval systems, since they allow to handle simple queries on content and structure. However, a fully-fledged XML retrieval system provides a lot more flexibility and completeness with respect to the formulation and execution of structural queries. Earlier approaches to structured retrieval

by Burkowski (1992) and Navarro and Baeza-Yates (1995) already considered most of the functionality that is expected from current systems working with XML data. Structured retrieval enables to freely compose queries with *content* and *structure* conditions. We can ask for instance for sections about "XML retrieval" inside documents about "text retrieval", assuming that sections and documents are tagged in the collections as in the example in Figure 1.1. In contrast to fielded search, which only allows to restrict the term query to certain fields of a document, structured retrieval allows to express any containment relation of structure elements and terms, like the request of relevant sections being contained in certain documents. Furthermore, the shown structured query also states directly the desired ranked output element, here sections instead of documents.

With the omnipresence of XML data as the mark-up language for machine-readable structure, "structured retrieval" became "XML retrieval" with special query languages designed to express structural requests on XML like XQuery Full-Text (Amer-Yahia et al., 2007) or NEXI (Trotman and Sigurbjörnsson, 2004). The latter is designed in close connection to ongoing research efforts in the area of XML retrieval brought together by the INEX evaluation initiative (Malik et al., 2007).

XML retrieval does not require the user to specify at indexing time fields of interest, but allows to query the content of any tagged fragment of the collection. These features asks for different index designs. When every possible tag can be queried, an inverted document index regarding each tag as a single document becomes highly redundant. Each level of nesting causes repetition of its content.

Entity Retrieval Entity retrieval sets the focus level of retrieval one more step higher. It allows to search and rank named entities included in any kind of text sources. We could ask such a system to list persons, dates and/or locations with respect to a given query topic. An entity retrieval request, looking for persons associated with user studies on XML retrieval might return among others the gray-shaded person entities in Figure 1.1, if the outlined document belongs to the considered text collection. Document borders should not play any role here. Multiple mentions of a specific entity can be extracted from multiple documents, but the same entity should be listed only once in a ranked result list. Entity retrieval systems are useful to provide a very condensed mind-map-like overview on a given topic. One could also filter out a specific entity type to get a ranking on set of "candidate" entities, like employees in *expert search*. The very focused entry level comes with the disadvantage, that relevance is less clear to verify. A user cannot

simply check the relevance of a returned entity without seeing the context it is mentioned in. In the same way, retrieval systems cannot rank entities directly, but have to rank text fragments and propagate their relevance appropriately to the included entities. Entity retrieval also relies heavily on the availability and accuracy of natural language processing (NLP) tools, needed for the correct recognition and classification of named entities within the text corpus. In the visualized example document (Figure 1.1), NLP tools are thus responsible for the correct gray-shading of names and dates.

The notion of "entity retrieval" was introduced recently, however, earlier work considers typical cases of entity ranking as for instance expert search (Balog et al., 2006) or factoid and list queries in question answering (Voorhees and Dang, 2005). Chakrabarti et al. (2006) already abstracts from a domain specific solution and describes a system that can rank any type of entities by proximity features. Also Hu et al. (2006) describes person and time search as two instances of the more generic entity retrieval task.

Question Answering On the way towards focused retrieval answers, it is important to mention question answering systems as well. However, they remain somewhat outside the presented line from document to entity retrieval, since their emphasis lies on understanding the semantics of a (natural language) query, rather than on the ranking task itself (Radev et al., 2002; Lin and Katz, 2003). Still the connection of question answering to the other introduced focused retrieval tasks is strong. Once a query is analyzed, the system searches for sentences or parts of sentences that state the wanted answers. Question answering could be seen as sentence retrieval in that case. Whenever faced with a simple fact query, asking for example for a person or location, systems might even use entity ranking techniques and output the requested entity only. Most research on question answering systems was driven by the corresponding track of the TREC evaluation initiative (Dang et al., 2006; Hovy et al., 2000). Question answering further shares with entity ranking the dependence on NLP tools. They are used here first of all on the query to determine its target (fact, relation, etc.), but later also on the retrieved sentences to select those stating an answer to the query. In fact, question answering goes here a step further than other ranking tasks, since it typically selects the best matching item only to present it as an answer to the user.

This work picks out document, XML, and entity retrieval – thus 3 different granularity levels – on which retrieval techniques are presented with respect to effectiveness and/or efficiency. Others, like passage and fielded retrieval are partly covered by XML retrieval methods as well, though it will not

be explicitly mentioned in the respective places. Only question answering remains out of scope, as far as it concerns the semantical analysis of the query.

1.2 Adaptivity in Text Search

Information retrieval research tried over decades to improve search precision by introducing new retrieval models and tuning the existing ones. Those models applied to ad-hoc retrieval tasks rank a collection of documents given a set of keyword terms. However, we can often observe that such simple keyword queries are not appropriate to express real information needs. Whereas some search tasks have characteristic and meaningful keywords, others cannot be expressed that way, or at least the user is not aware of those keywords. Precision gain is here easier to achieve by further *adaptation* of the search process. Adaptation here simply means to influence the retrieval result by other means than adding or removing single search keywords. The underlying hypothesis is that users typically underspecify their information need while formulating a search query. Next to the explicitly stated keywords users often have further constraints to their search. To take these constraints into account, retrieval systems have to become adaptive to a set of user parameters.

User Parameters in Information Retrieval Some introductory examples will illustrate what kind of parameters adaptive text search has to consider:

- Instead of returning the lengthy text of the European "constitution", a citizen interested in the election of the European parliament might be more satisfied by getting just a small relevant section about the voting system. Thus the *granularity* of answers needs to be scalable. Furthermore, depending on the *level of expertise* of the searcher, either the original law text or a simplified better understandable version will be highly appreciated here.

- Having a latex allergy and looking for information about these materials on the web, a physician will not be pleased getting information about excellent text-layout systems. In this case the *topicality* of the query is not covered by the query words alone and needs the adaptivity of the system.

- Searching for the best price of a new camera, we are not interested to see, how much cheaper consumer electronics are in low-tax countries.

Here, the *locality* of the query plays an important role. Furthermore, we are definitely not interested to see outdated old price lists. So, also the *temporality* constraints play a role here. In case we know more about the *structure* of typical results, it might also be beneficial to express a preference of table-like price lists over plain text.

- If the same person, on the other hand, wanted to compare product reviews on certain cameras, she/he does not like to find only special product offers in the ranked list. Here, the *genre* constraint is missing in the query. It might help to add the word "review" to the set of search terms, but in the same way it can cause other relevant pages to disappear, since they do not mention the new keyword, but write about the products.

The examples mention several dimensions of meta-constraints for the search process, namely: (1) topicality, (2) genre, (3) temporality, (4) locality, (5) required level of expertise, (6) structure, and (7) the granularity of the wanted results. The given list might not be complete, but it covers many aspects that play a role in text search.

It is important to notice how the parameters differ in type. Whereas we distinguish for topicality and genre usually a limited set of different topics or genres, time is measured on a continuous scale and especially the locality parameter often even needs to consider different levels of accuracy. Also the documents themselves can often not be classified clearly to belong to one or multiple topics, genres, or locations. It is more appropriate to speak of a graded rather than a binary classification. Correspondingly, users might want to express "hard" or "soft" search constraints. Either they want the retrieval system to strictly filter the results or they only state a preference for a certain class of documents.

Explicit vs. Implicit Adaptivity Another important question regarding adaptivity of retrieval is, whether the system automatically tries to detect the user's working context and adapt the search appropriately or whether the user should state search constraints explicitly on his/her own. Both approaches come with advantages and disadvantages. Explicit feedback approaches ask for more input from the user, therefore they require more of the user's attention and time. Moreover, additional feedback often needs special user interfaces to enable the user to express further search constraints. Explicit adaptivity also assumes that users have the necessary knowledge of their search topic to answer feedback questions appropriately. Implicit approaches, however, rise the question how the user's context can be derived

automatically. In general, this task is rather difficult and in many cases even impossible. Automatic context detection is furthermore error-prone. It might sense a situation incorrectly and filter out results someone wanted to see. If a searcher is not aware of the applied (wrong) search adaptation, or unable to correct constraints in the way wanted, she/he even feels loosing control of the system.

Search Process Adaptivity Whereas all previously considered forms of adaptivity still assumed a static search process, consisting of an initial query and a certain number of refinement steps, we can also seek after adaptivity in the interaction between user and system during the search. A system might for instance react to a given user query by asking clarification questions if necessary. The envisioned retrieval system would analyze a user query, recognize whether a query is still ambiguous, and knows how to ask for suitable feedback. Such a form of adaptivity combines in a way the explicit and implicit approaches. It proactively asks for clarification whenever a user query remains ambiguous, it can even suggest probable and effective further constraints, but it expects the user to give feedback and keeps her/him in control.

This thesis is concerned with most of the introduced aspects of text search adaptivity. With respect to the user parameters, the first chapter proposes an open approach that allows to incorporate multiple different meta constraints to a given keyword query. It also suggests a new type of explicit feedback. Further chapters concentrate on the case, when only parts of documents should be retrieved. In terms of adaptivity, XML and entity retrieval allow to express constraints on structure and the granularity of retrieval. In both cases, we consider only explicit forms of search constraints expressed in the query language. However, this is not necessarily meant as a restriction, but simply results from the fact, that prediction techniques for setting appropriate structural and granularity constraints do not exist yet.

1.3 Research Objectives

The work presented in this thesis is driven by a number of quite different research objectives. We will show connections between the different topics the thesis deals with in the introductory sections of all chapters as well as in the final review and outlook.

The first approached aim is the incorporation of user parameters into the text retrieval process. Suppose we know more about the user's working

context, when she/he issues a search by a simple term query, we would like to take this additional information into account for improving the retrieval results. Since context information is a rather broad term, which can be assigned to everything describing the situation of a user, it is interesting to investigate which dimensions of context information are useful to achieve more precise retrieval results. Several questions and tasks arise, along the line of this aim. In order to make effective use of context information, it is important

(A1) to model the information in an appropriate – preferably generic – way, that allows to score documents against the context information,

(A2) and to examine how to combine the relevance evidence with respect to the context model with the relevance based on the initial term query.

The mentioned research objectives assume knowledge about the search context. However, gathering knowledge about the user's working context is a problem in itself. A typical approach to achieve context information is the use of explicit or implicit feedback as described in the last section. The arising question is then:

(A3) How can we automatically detect and suggest effective search constraints for feedback?

When the user is allowed to constrain a search also by structural features, it is first of all important to find a suitable language to express queries on content and structure. Existing languages are either rather complex and hard to use and to implement or deliberately simplistic, limiting the expressive power more than desirable. From a system's point of view, we see several further issues when performing structured retrieval:

(B1) Common inverted indices are not appropriate for structured retrieval with a high level of nested elements. It is thus important to develop a new type of index that overcomes the high redundancy.

(B2) The basic operations of structured retrieval – first of all the evaluation of the containment condition – need not only support from the index, but also efficient algorithms for their execution.

(B3) Structured retrieval opens new possibilities for query optimization, which need to be analyzed.

Once having an efficient XML retrieval system and NLP taggers, that are able to recognize and classify named entities as well as the basic syntax of

sentences, we are able to work with text corpora coming with large amount of structured annotation data. The question then arises what new type of text search activities are possible using such a system and data. In other words, can we develop a framework that is adaptive to new type of retrieval tasks dealing with the search on entities, e.g. expert search, or the retrieval of dates to construct chronological timelines of events or the biography of a person. Such a framework needs mainly to address the question how we can rank entities, preferably by a generic approach that can be applied to different entity retrieval tasks. Since entities cannot be ranked directly by their text content, it is important

(C1) to model the relation between entities and texts that mention the entities,

(C2) and to develop and test relevance propagation models, that allow to derive the relevance of entities from related texts.

While the incorporation of context parameters in document retrieval models deals highly with *score combination*, the retrieval tasks on finer result granularity are more concerned with *score propagation*. Especially for entity retrieval we need to study models of score propagation in order to transfer the relevance evidence of different pieces of text towards the mentioned entities, since they cannot be scored directly. In this respect, XML retrieval stays right in the middle of the other two. It makes use of both score combination and propagation as its basic operators.

Thesis Outline The structure of this thesis directly follows the title "from document to entity retrieval" and divides the research work into three main chapters that examine text search on different levels of retrieval granularity:

(1) document retrieval,
(2) XML retrieval,
(3) entity retrieval.

The first chapter examines the refinement of document retrieval by context information. It thereby addresses the research questions (A1)-(A3). The following chapter on XML retrieval is more concerned with the systems efficiency as mentioned by the issues (B1)-(B3). Finally, the last chapter presents a framework for graph-based entity ranking that is mainly driven by the research goals (C1) and (C2).

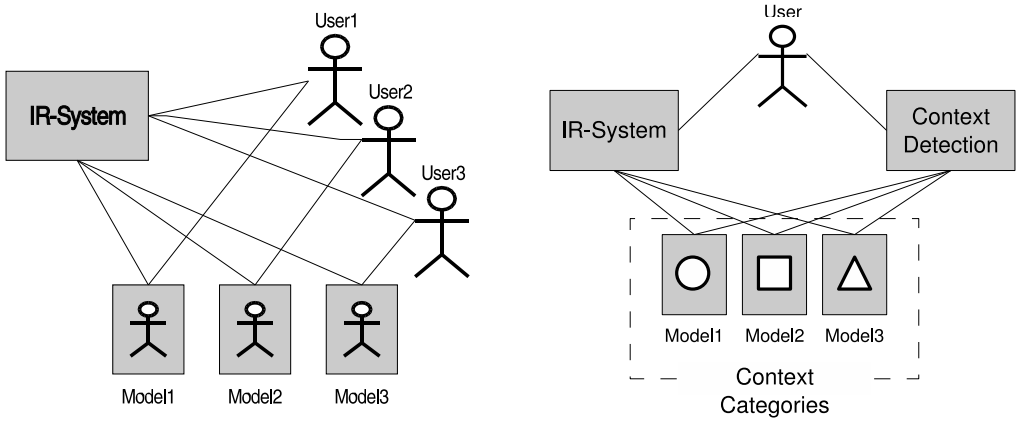
2

Context Refined Document Retrieval

Noticing that humans are thinking about, searching for, and working with information highly depending on their current (working) context, leads directly to the hypothesis that retrieval systems could improve their quality by taking this contextual information into account.

A user's information need is only vaguely described by the typical short query, that the user expresses him/herself to the system. There are at least two reasons for this lack of input precision. First of all, users who search for a certain piece of information have incomplete knowledge about it themselves. The difficulty to describe it is thus an immanent problem of any information need and hardly to overcome. A second reason for insufficient query input, however, touches the area of context information and might in principle be easier to address. Although a human's search context provides a lot of information about his/her specific information need, a searcher is often not able and not used to explicitly mention it to a system. When asking another human instead of a system, the counterpart would be able to derive implicit contextual information him/herself.

We first address the question how the already available information about the user's context can be employed effectively to gain highly precise search results. This part is based on earlier published work (Rode and Hiemstra, 2004). Later we show how such meta-information about the search context can be gathered. The latter is presented also in the two articles (Rode et al., 2005; Rode and Hiemstra, 2006).



(a) User-Dependent Models (b) User-Independent Models

Figure 2.1: Context Modeling: User vs. Category Models

2.1 Context Modeling for Information Retrieval

Aiming at a context-aware text retrieval system, we first have to investigate how context can be modeled appropriately so that an IR system can take advantage of this information. One of the first upcoming matters will probably be described by the following question: Should we try to build a model for each individual user or should we classify the user with respect to user-independent predefined context-categories? Both kind of systems are outlined in Figure 2.1. We will choose the latter option, but first discuss the advantages and disadvantages of both by pointing to some related research in the respective areas.

User-Dependent Models A first and typical example for this approach is shown by Diaz and Allan (2003). The authors suggested to build a user preference language model from documents taken out of the browsing history. Since the model reflects the browsing behavior of each individual user, it describes his/her preferences in a very specific way.

However, humans work and search for information often in multitasking environments (Spink et al., 2006). Thus, their information need changes frequently, often without overlaps between different tasks. A static profile of each user is not appropriate to take into account rapid contextual changes. For this reason, Diaz and Allan (2003) also tested the more dynamic version of session models derived from the most recent visited documents only. With the same intention, Bauer and Leake (2001) introduced a genetic "sieve" algorithm, that filters out temporally frequent words occurring in a stream

of documents, whereas it stays unaffected by longterm front-runners like stop words. The system is thus aware of sudden contextual changes, but cannot come up directly with balanced models describing the new situation.

Summarizing the observations, individual user models enable a more user specific search, but either lack a balanced and complete modeling of the users interests or remain unaware of alternating contexts.

User-Independent Models Although context itself is by definition user-dependent, it is possible to approximately describe a specific situation by selecting best-matching pre-defined concepts, that are themselves independent of any specific user. A concept in this respect might range from a subject description (e.g. "Music") to geographical and temporal information (e.g. "the Netherlands", "16th century"). To introduce a clear terminology, each *concept* belongs to a *context dimension*, like subject, genre, or location, and characterizes a *category* of documents.

The evaluation initiative TREC (Text REtrieval Conference) had a special track that addresses user feedback and contextual meta-data. The setting of the so-called HARD track (High Accuracy Retrieval from Documents) is typical for this type of user-independent context modeling (Allan, 2003, 2004). Along with the query, a set of meta-data concepts characterize the context of each specific information need. The HARD track considers thereby the context dimensions: familiarity, genre, subject, geography, and related documents. Apart from the related documents, all dimensions come with a predefined set of concepts. It is then suggesting to build models that classify documents according to each of these concepts.

Following this approach of context modeling, it needs to be explained where the additional context meta-data comes from. Whereas Belkin et al. (2003) preferred to think of it as derived by automatic context-detection from the users' behavior, He and Demner-Fushman (2003) described the collecting of contextual information in a framework of explicit negotiation between the search system and the user. Further experiments in this area are presented by Sieg et al. (2004a). The authors tried to employ a conceptual hierarchy of subjects, as established by the "Open Directory Project"[1] or "Yahoo"[2], as contextual models. In a first experiment, queries were compared to these concepts and the best-matching subjects were displayed to the user for explicit selection. In order to avoid this negotiation process, long-term user profiles were introduced for automatic derivation of matching subjects, which cluster the former interests of the user in suitable groups. However,

[1] see `http://www.dmoz.org`

[2] see `http://dir.yahoo.com`

these user-dependent models suffer from the same limitations as mentioned before.

Although automatic context detection is problematic, user-independent context modeling comes up with a number of advantages:

- Whereas user modeling suffers often from sparse data, conceptual models are trained by all users of the systems and therefore will become more balanced and complete.
- Conceptual models do not counteract the search on topics entirely new to the user. A user dependent model is always based on the search history and therefore supports the retrieval of related items, but counteracts the search on new topics.
- Assuming a perfect context detection unit, the search system can react more flexible with respect to a changing context of a user.
- New users can search efficiently without the need to train their user preference models in advance.
- It is theoretically possible to switch back anytime from automatic context detection to a negotiation mode, which enables the user to control the system effectively.

Taking a closer look on conceptual context modeling, the first task will be to identify appropriate categories of the users situation with respect to the retrieval task. Whereas we can call almost everything surrounding the user as context, we only need those data that allows to further refine the information need of the user. The context dimensions and concepts used by the HARD track obviously allow to refine the search space, but they are not the only appropriate ones. We can easily extend this set by other dimensions like language or time/date.

One might notice that the dimensions suggested so far originate more from a document than from a user centered view. Since we want to fine-tune the retrieval process, it is handy to have categories that directly support the document search. However, starting from the users context, this already requires a first translation from context description to document categories. For instance, the situation of a biology scientist sitting at his work might be translated to the following context description: familiarity with search-topic: "high", search genre: "scientific articles", general subject: "biology". The translation of the user's situation into the desired context categorization is, of course, an error-prone process. Thus, the possibility to allow the user to explicitly change the automatically performed categorization of his/her context will be an important issue.

2.1.1 Conceptual Language Models

The retrieval process itself is enhanced by multiple text-categorizations based on the selected concepts that match the users' situation. Thus, the retrieval system needs to maintain models for each context concept that can be used as classifiers, e.g. a model for *scientific articles* should be applicable to filter out scientific articles from an arbitrary set of documents.

Looking at the HARD track experiments of other groups, e.g. at the work of Belkin et al. (2003) or Jaleel et al. (2003), every context dimension is handled with different techniques ranging from a set of simple heuristic rules as used for classifying the genre to applying algorithms like Gunnings "Fog Index" measure (Gunning, 1968) to rate the readability. The techniques might enable an IR system to utilize the specific given meta-data, but the approaches lack a uniform framework that enables extending the system to work with other meta-data categories as well.

Instead of introducing another set of new techniques, we suggest to apply statistical language models as a universal representation for all context categories that are not directly supported by existing document meta-data (documents in the HARD collection contain publishing dates for instance). Obviously, language models can be utilized effectively as subject classifiers, but we think, it is also possible to use them to judge about the genre or readability of a document. In the latter case, we can for instance assume that easily readable articles will probably consist of common rather than of special terms. For geography models, on the other hand, we would expect a higher probability to see certain city names and persons, whereas genre models might contain often occurring verbs or a differing number of adjectives. Unfortunately, the envisioned uniform handling of all context dimensions could not be tested sufficiently with the given collection, query set, and meta-data of the HARD track. The provided query meta-data specifies one of the predefined concepts for each context dimension, or leaves a context dimension unrestricted without specification. The latter happened more often when a context dimension was considered as not helpful on the collection. The used corpus of newspaper data for instance does not show enough heterogeneity for distinguishing genre or readability and the two considered location concepts "US" and "non-US" have been too broad for suitable query restriction. Still, the uniform classification approach forms the background of our following considerations.

In order to enable context-aware query refinement, it is therefore sufficient to enhance the retrieval system by a set of language model classifiers for each context category. The remaining task to perform all document classifications and to combine them for a final ranking according to the entire search

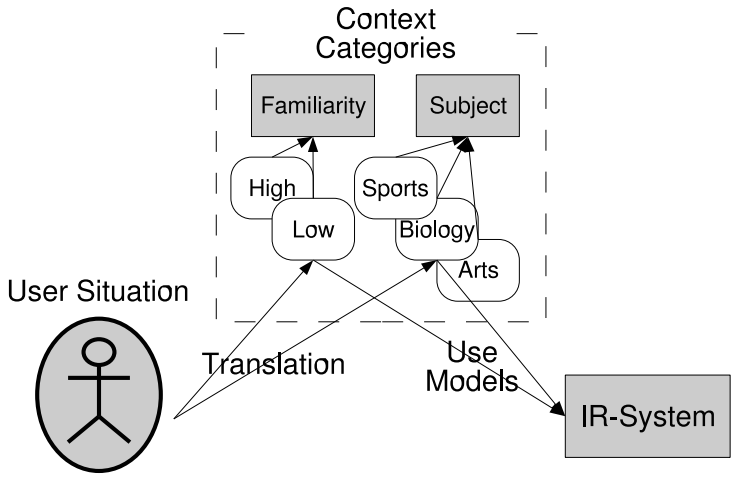


Figure 2.2: Context Modeling with Conceptual Language Models

topic will be addressed in the next section. Figure 2.2 sketches roughly the described system.

Learning Application An IR system working with conceptual models will profit from being a self-learning application. While it is necessary to start the system with basic models for each concept, it is beneficial to have the system training its models by the feedback of the user in the later phase of use.

Anytime a user indicates (explictly or observed by her/his browsing behavior) that a certain document matches her/his information need, we can assume that it also matches the selected conceptual models. Therefore, the content of such a document can be used to train the context models. In the setting of the HARD track we can use the relevance assessments of the training topics to improve our models in the same way.

2.2 Ranking Query and Meta-query

If concept language models are available that describe the user's context, further on called *meta-query* models M_i, we are able to classify the documents according to each single context dimension, but we need to come up with a single final ranking including every single source of relevance evidence. There are basically three options to perform this task:

- *Query Expansion* in order to build one large final query that considers the initial query as well as all meta-query models,

- *filtering* of the results according to each classifier,
- *score combination* in order to aggregate the scores of single classifications.

Using query expansion techniques would lead to the difficult task to select a certain number of representative terms from each model. Since the query and "meta-query" models differ highly in length, we cannot simply unite all terms to one combined query. Filtering, on the other hand, only allows black-and-white decisions for or against a document. However, thinking of a query refinement on several context dimensions, it is likely that a document is judged relevant by a user even if it does not match all of the associated classifiers. Therefore, we opt here for a combined ranking or re-ranking solution, which allows to consider each context-classification step adequately.

2.2.1 Combined Ranking of Query and Meta-Query

For discussing the ranking of documents according to the query and meta-query we first introduce some common notation. Let the random variables Q, D denote the choice of a query, respectively document, and $r/\bar{r}$ mark the event, that D is regarded as relevant/not relevant. Further, M represents in our case the *meta-query*, consisting of several single models M_i for each context concept involved :

$$M = \{M_1, M_2, \ldots, M_n\}.$$

Using the odds of relevance as a basis, we can deduce it to probabilities that we are able to estimate. Q and M are assumed to be independent given D and r:

$$\begin{aligned}
&\frac{P(r|Q,M,D)}{P(\bar{r}|Q,M,D)} = \frac{P(Q,M,D|r)P(r)}{P(Q,M,D|\bar{r})P(\bar{r})} = \frac{P(Q,M|D,r)P(D|r)P(r)}{P(Q,M|D,\bar{r})P(D|\bar{r})P(\bar{r})} \\
= \;& \frac{P(Q,M|D,r)P(r|D)}{P(Q,M|D,\bar{r})P(\bar{r}|D)} = \frac{P(Q|D,r)P(M|D,r)P(r|D)}{P(Q|D,\bar{r})P(M|D,\bar{r})P(\bar{r}|D)} \\
\propto \;& \frac{P(Q|D,r)}{P(Q|D,\bar{r})}\frac{P(M|D,r)}{P(M|D,\bar{r})} \propto \log\left(\frac{P(Q|D,r)}{P(Q|D,\bar{r})}\right) + \log\left(\frac{P(M|D,r)}{P(M|D,\bar{r})}\right).
\end{aligned}$$

The prior document relevance $P(r|D)/P(\bar{r}|D)$ is dropped from the equation in the third row. We assume that there is no a-priori reason that a user would like one document over another, effectively making the prior document relevance constant in this case.

The simple derivation now allows to handle query and meta-query separately but in a similar manner. In terms of the user's information need we can regard Q and M as alternative incomplete and noisy query representations.

Combining the resulting document rankings from both queries gathers different pieces of evidence about relevance and thus helps to improve retrieval effectiveness (see e.g. Croft, 2002).

The remaining probabilities can be estimated following the language modeling approach. In particular, we will use a language modeling variant shown by Kraaij (2004), which directly estimates the above required logarithmic likelihood ratio $LLR(Q|D)$:

$$\begin{aligned} LLR(Q|D) &= \log\left(\frac{P(Q|D,r)}{P(Q|D,\bar{r})}\right) \\ &= \sum_{t\in Q} |t \; in \; Q| * \log\left(\frac{(1-\lambda)P(t|D)+\lambda P(t|C)}{P(t|C)}\right). \end{aligned}$$

The probability of a term given an irrelevant document $P(t|D,\bar{r})$ is estimated here by the collection likelihood of the term $P(t|C)$. The smoothing factor λ interpolate document and collection likelihood.

Since we want to relate the scores of the query and meta-query to each other, we have to ensure that their probability estimates deliver "compatible" values (Croft, 2002). Especially query length normalization plays a crucial role in this case. Notice, that Q and M differ widely with respect to their length. Thus, a simple LLR-ranking would produce by far higher values when it is applied to the meta-query. Using $NLLR$ instead, a query *length normalized* variant of the above measurement, helps to avoid score incompatibilities:

$$NLLR(Q|D) = \sum_{t\in Q} P(t|Q) * \log\left(\frac{(1-\lambda)P(t|D)+\lambda P(t|C)}{P(t|C)}\right).$$

A slightly modified but order preserving version comes with the desirable property to assign zero scores to all irrelevant documents and positive scores to all documents that contain at least one of the query terms:

$$\begin{aligned} NLLR(Q|D) &\propto \sum_{t\in Q} P(t|Q) * \log\left(\frac{(1-\lambda)P(t|D)+\lambda P(t|C)}{\lambda P(t|C)}\right) \\ &= \sum_{t\in Q} P(t|Q) * \log\left(\frac{(1-\lambda)P(t|D)}{\lambda P(t|C)}+1\right). \end{aligned}$$

Whenever we refer in the following to the NLLR for experiments, we mean in fact this modified calculation.

Ranking according to the Meta-Query As mentioned above, we would like to rank documents according to query and meta-query in the same way. However, since M consists of several single language models $M_1, \ldots, M_n$ we need to take a closer look to this matter as well.

If M is substituted by $M_1, \ldots, M_n$, the resulting equation can be factorized, given the independence of $M_1, \ldots, M_n$:

$$\log\left(\frac{P(M_1,\ldots,M_n|D,r)}{P(M_1,\ldots,M_n|D,\bar{r})}\right) = \log\left(\frac{P(M_1|D,r)}{P(M_1|D,\bar{r})} * \ldots * \frac{P(M_n|D,r)}{P(M_n|D,\bar{r})}\right)$$
$$\simeq \frac{1}{n}\sum_{i=1}^{n} NLLR(M_i|D).$$

Using the length-normalized $NLLR$, the second line of the equation is strictly speaking not proportional to the first one, however we argued before why the length normalization is necessary here. The second line of the equation also introduces a second type of normalization. The factor $\frac{1}{n}$ is used to ensure that the final score of the meta-query does not outweigh the score of the initial query. Especially if the number n of context dimensions is growing, not only the overall score of the documents would increase, but also the entire meta-query would get a higher weight than the initial term query.

A last remark concerns the choice of the smoothing factor λ. In contrast to typical short queries, the role of smoothing is less important here, since we can assume that the model is a good representation of relevant documents and therefore contains most of their words itself. We thus argue to use a smaller value for λ here than in case of the query ranking to stress the selectivity of the models.

2.3 Experiments

The experiments in this section test the usage of context meta-data on the retrieval quality applying the proposed score combination approach.

As mentioned already, we experimented in the setting of TREC's HARD track, in this case with the collection and topic set from 2004. The collection consists of 1.5 GB of news papers data including articles from 8 different news papers from the year 2003. The query set contained 50 topics described by title, description, and narrative as standard for most TREC evaluations. Furthermore, each topic comes with a set of associated meta-data concepts considering the dimensions familiarity, genre, subject, geography, and related documents. The judgments from the assessors consider 3 different cases. In contrast to the binary relevance decision the assessors could mark whether a document is relevant to the topic only or relevant with respect to topic and

query meta-data. Correspondingly, the evaluation distinguishes so-called *soft* and *hard* relevance. The first considering both types of relevance, the later more strict evaluation regards only those documents as relevant that match topic and meta-query.

Collecting Data for the Models We have used only a part of the meta-data that came along with the queries, namely the *subject*, *geography* and *related text* sections. Having appropriate models at hand is a crucial requirement for any kind of experiments and the need to construct them ourselves has led to this limitation.

The *subject* data was chosen, because it was considered to work best with respect to the purpose to classify texts. It is probably easier to identify sport articles by their typical vocabulary then to distinguish between genres. *Geography* data, on the contrary, can be regarded as a less typical domain for applying language model classifiers. And finally *related text* documents were used to demonstrate their straightforward integration in the proposed context modeling framework. We built a unified language model from all related text sources and used it simply as another meta-query model M_i in the scoring procedure.

In order to construct language models for subject classification, we used three different sources of data:

- manual annotation,
- APE keywords (see explanation below),
- and the training data.

Firstly, we manually annotated 500 documents for each chosen subject among the queries, e.g. sports, health and technology. The 500 documents have been preselected by a simple query containing the subject term and additional terms found in a thesaurus. The aim of this step was to detect 150-200 relevant documents as a basic model representing its subject. For construction of a language model all terms occurring in those documents were simply united to build one large "vocabulary" and probability distribution.

Although the number of documents might look appropriate for building a basic text classifier, the way we gathered the documents cannot ensure the models to be unbiased. In order to further improve the models, we used the keyword annotation coming along with the documents. During the manual classification process we observed that the keyword section of documents from the Associated Press Newswire (APE) provide very useful hints and in many cases HARD subjects can easily be assigned to APE keywords. It seemed admissible from research perspective to exploit this information as

		title only		title + desc		all	
		Base	Meta	Base	Meta	Base	Meta
soft	MAP	0.177	0.214	0.219	0.303	0.271	0.361
	R-Prec	0.211	0.255	0.245	0.335	0.308	0.374
hard	MAP	0.192	0.226	0.220	0.302	0.269	0.346
	R-Prec	0.206	0.244	0.214	0.298	0.294	0.349

Table 2.1: MAP and R-Precision for Baseline and Meta-data Runs

long as we restrict it to a small part of the corpus, in this case APE news only. However, since HARD subjects cannot be mapped one-to-one to APE keywords, our subject models differed considerably afterwards in length and quality. For the geography models, the link between query meta-data and document keywords was easier to establish. Therefore, the geography models highly benefit from using the keywords.

In a last step, we automatically enhanced the models by data obtained from the annotated training topics as mentioned above (see Section 2.1.1). If any document was judged as relevant to a specific training query, this also means that the document matches all the meta-data constraints of that query. Thus, all relevant documents belonging to a query asking e.g. for sport articles, apparently are sport articles themselves, and can therefore be used to enrich the sport articles model.

Baseline Runs Every HARD track topic is specified by a title, description and topic-narrative section, which could be used for the baseline runs. The most realistic scenario would be to use only the short title queries, since users – at least on the web – express their information needs typically by a few keywords only. In order to examine the influence of the initial query length to improvements made by context meta-data, we also compute runs based on the union of terms in the title and description fields, respectively using the terms from all 3 fields (see Table 2.1). The expectation here would be that meta-data especially helps short user queries, rather than well-described information needs. All three baseline runs were ranked according to $NLLR(Q|D)$.

Meta-data Runs Corresponding to the baseline runs, three further runs were calculated that make use of several dimensions of meta-data. The scores of the initial query and meta-query were combined here as shown in the Section 2.2. We took here the following meta-data dimensions into account: subject, geography, and related texts as $M_1 \ldots M_3$. Table 2.1 gives

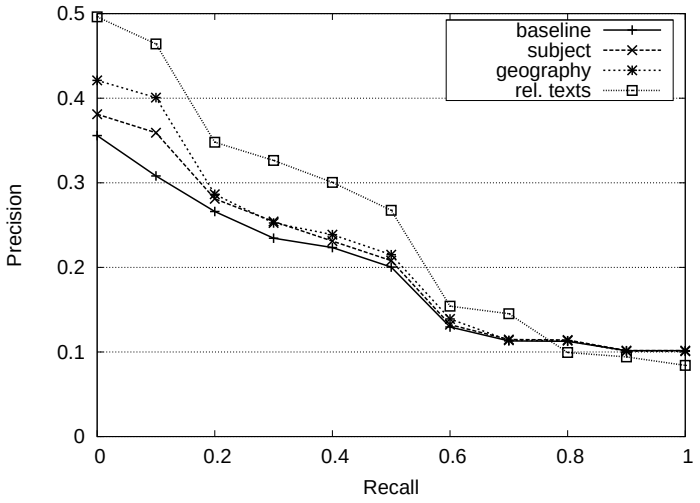


Figure 2.3: Comparing Precision/Recall for each single Meta-data Category

an overview on the achieved mean average precision (MAP) and the average R-Precision of all runs, the latter being the official measure in the HARD track evaluation. The result overview shows first of all that our approach for handling contextual data is able to improve retrieval results, for soft as well as for hard relevance. We expected higher relative improvements when using context information together with short user queries, however, our results show that long queries still can profit in the same way from contextual data. Furthermore, evaluation against hard or soft relevance shows nearly the same improvements. The interpretation here is less obvious. We might have expected improvements mainly for the evaluation against hard relevance, since it considers only documents matching the meta-query requirements. Instead, the evaluation with respect to soft relevance holds the same improvements. The outcome indicates that query and meta-query are less independent than assumed by the ranking model. Both are not orthogonal constraints of the underlying information need, but the meta-query supports and refines the initial term query.

We performed further experiments to find out if the given context dimensions are equally useful for improving the system performance. Figure 2.3 presents the resulting precision-recall graph if the queries are associated with only one dimension of meta-data. It considers title and description queries and hard relevance only. In order to get comparable results for all dimensions, we needed to restrict the evaluation to a small subset of 11 topics that came with geographical and subject requirements we could support with appropriate models. For instance, we dropped topics asking for the subject

society, since the associated classifier was considered rather weak – based on a considerable fewer number of documents – compared to others. Such a restriction is admissible, since we were interested in the retrieval improvements in the case appropriate models are available, however, the remaining topic set was unfortunately a relative small base for drawing strong conclusions.

The graph suggests that the utilization of geography and subject preferences allow small improvements whereas related texts considerably increase the retrieval quality. In fact, using related text information alone shows even better results than its combination with other meta-data. As a conclusion, it might be interesting to test in further experiments if a more parameterizable approach that can assign different weights to each context dimensions is able to prevent such negative combination effects. However, a large set of parameters that needs training to be set appropriately should be avoided in principle. The displayed graph shows further that the usage of contextual information especially enhances the precision at small levels of recall, which meets perfectly the "high accuracy" aim of the approach.

2.4 Interactive Retrieval

When information retrieval left the library setting, where a user ideally could discuss her/his information need with a search specialist at the help-desk, many ideas came up how to imitate such interactive search scenario within retrieval systems. Belkin (1993), among others, broadly sketches the system's tasks and requirements for interactive information seeking. We do not want to further roll up the history of interactive information retrieval here, but to remind briefly its main aims.

In order to formulate clear queries, resulting in a set of useful, relevant answers, the user of a standard information retrieval system needs knowledge about the collection, its index, the query language and last but not least a good mental model of the searched object. Since it is unrealistic to expect such knowledge from a non-expert user, the system can assist the search process in a dialogue like manner. Two main types of interactive methods try to bridge the gap between a vague information need and a precise query formulation:

Relevance Feedback Giving feedback helps the user to refine the query without requiring sophisticated usage of the system's query language. Query terms are added or re-weighted automatically by using the relevant examples selected by the user (Salton and Buckley, 1990; Harman, 1992). The examples shown to the user for judgment can either be documents, sentences out

of those documents or even a loosely bundle of terms representing a cluster of documents. Experiments within TREC's interactive HARD track showed many variants of such techniques (Allan, 2003, 2004). By presenting example answers to the user, relevance feedback can also refine the user's mental image of the searched object.

Browsing Techniques subsumed by the keyword "browsing" provide an overview on the existing document collection and its categorization as for instance in the "Open Directory Project"[3], or visualize the relation among documents (Godin et al., 1989). The user can restrict the search to certain categories. This can also be regarded as a query refinement strategy. It is especially helpful, when the selected categorical restriction cannot be expressed easily by a few query terms.

The query clarification technique, we are proposing in the following, belongs mainly to the first type, the relevance feedback methods. However, it combines the approach with summarization and overview techniques from the browsing domain. This way it tries not only to assist formulating the query, but also provides information about the collection in a query specific preview, the so-called *query profile*. Following an idea of Diaz and Jones (2004) to predict the precision of queries by using their temporal profiles, we analyzed the application of different query profiles as an instrument of relevance feedback. The main aim of the profiles is to detect and visualize query ambiguity and to ask the user for clarification if necessary. We hope to enable the user to give better feedback by showing him/her this summarized information about the expected query outcome.

2.4.1 Related Approaches

In order to distinguish our approach from similar ones, we take a look at two comparable methods. The first one is a search interface based on clustering suggested by Palmer et al. (2001)[4]. It summarizes results aiming at query disambiguation, but instead of using predefined concepts as we suggest for our topical profiles, it groups the documents using an unspecified clustering algorithm. Whereas the clustering technique shows more topical adaptiveness, our static categories are always based on a meaningful concept and ensure a useful grouping.

[3] see `http://www.dmoz.org`

[4] The one-page paper briefly explains the concept also known from the *Clusty* web search engine (`http://clusty.com`) coming from the same authors.

Another search interface proposed by Sieg et al. (2004b) assists the user directly in the query formulation process. The system compares the initial query with a static topic hierarchy and presents the best matching concepts to the user for selecting preferences. The chosen concepts are then used for query expansion. In contrast, our query profiles are not based on the few given query terms directly but on the results of an initial search. This way, we get a larger base for suggesting appropriate concepts and we involve the collection in the query refinement process.

The mentioned approaches exclusively consider the topical dimension of the query. We will further discuss the usage and combination of query profiles on other document dimensions, in this case temporal query profiles.

2.5 Query-Profiles

Looking from the system's perspective, the set of relevant answers to a given query is the set of the top ranked documents. This set can unfortunately differ greatly from the set of documents relevant to the user. The basic idea of query profiles is to summarize information about the system's answer set in a suitable way to make such differences obvious.

A *query profile* is the distribution of the top ranked documents in the result set along a certain property dimension, like time, topic, location, or genre. E.g. a temporal query profile shows the result distribution along the time dimension, a topical profile along the dimension of predefined topics the documents belong to.

The underlying assumption of the profile analysis is that clear queries result either in a profile with one distinctive peak or show little variance in case the property dimension is not important for the query. In contrast, we expect ambiguous queries to have query profiles with more than one distinctive peak.

Whereas the general ideas stay the same for all kinds of query profiles, there are several domain specific issues to consider. We will thus take a closer look on generating temporal and topical profiles, the two types used in the later experimental study.

2.5.1 Generating Temporal Profiles

Having a date-tagged corpus, a basic temporal profile for a given query is simple to compute. We treat the 100 top ranked documents D_j from the baseline run as the set of relevant answers and aggregate a histogram with

monthly time steps H_i:

$$H_i = |\{D_j | month(D_j) = i\}|.$$

The decision for the granularity of one month is based on the overall time span of the test corpus and the timeliness of news events. Other granularities, however, could be considered as well.

As a next step, we perform a *time normalization* on the profile. Knowing that the corpus articles are not evenly distributed over the total time span, the time profile should display the relative monthly frequency of articles relevant to the given topic rather than absolute numbers. Therefore, the frequency of each monthly partition H_i is divided by the total number of corpus articles C_i originating from month i. In order to avoid exceptionally small numbers, the averaged monthly corpus frequency $avg(C)$ is used as a constant factor:

$$H_i^* = \frac{H_i}{C_i} * avg(C).$$

Furthermore, we perform moving average smoothing on the histogram, a technique used for trend analysis on time series data (Chatfiled, 1984). It replaces the monthly frequencies of the profile by the average frequencies of a small time window around the particular month. We used here a window size of 3 months:

$$H_i^{**} = \frac{H_{i-1}^* + H_i^* + H_{i+1}^*}{3}.$$

The graph in Figure 2.4 shows an example of a resulting temporal profile. There are two reasons for using such a smoothing technique. First, the timeline the search topic is discussed in the news will often overlap with our casual monthly partitioning. Second, although we want to spot peaks in the profile, we are not interested in identifying a high number of splintered bursts. If two smaller peaks are lying in a near timely neighborhood they should be recognized as one.

Finally, we want to determine the number, bounds, and the importance of peaks in the temporal profile. Diaz and Jones (2004) tried several techniques for this purpose and decided to employ the so-called burst model from Kleinberg (2003). It assumes a hidden state machine behind the random events of emitting the specific word in certain frequencies. The assumed machine changes over time between its norm and peak state, corresponding to phases with normal and high emission of the word respectively. The aim is then to find the unknown state sequence with the highest probability to cause the observed random events of the time profile. Kleinberg employs for this task the Viterbi algorithm.

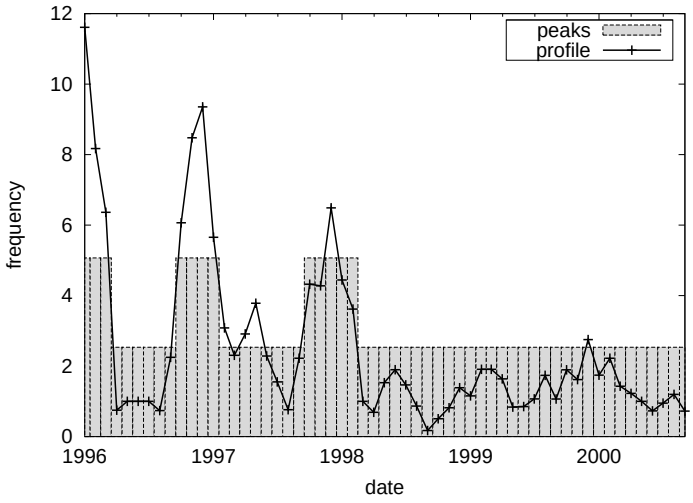


Figure 2.4: Temporal Profile of Topic 363: *Transportation Tunnel Disasters*

We have used for the generation of temporal profiles a two state automaton $\mathcal{B}^2_{1.5,0.02}$. The work of Kleinberg (2003) gives a detailed description of the automaton and its parameters. The considerably different setting of parameters – especially the very low value of $\gamma = 0.02$ – compared to Kleinberg's experiments can be explained by the fact that we analyzed profiles of word frequencies which are already averaged on the level of months. Hence bursts will remain smaller and less distinctive.

When we also want to compute a measure for the importance of the found peaks P_j, the corresponding frequency values of the temporal profile can simply be summed up. A further division by the average of such frequency sums $avg(P)$ leads to a value for peak intensity better comparable among different temporal profiles:

$$P_j = \sum_{i \in range(P_j)} H_i^{**}, \qquad intensity(P_j) = \frac{P_j}{avg(P)}.$$

2.5.2 Generating Topical Profiles

Generating topical profiles faces different issues than the ones explained for the temporal dimension. First and most important, the corpus is not topic-tagged. A topic classification is therefore required. Secondly, the topical dimension is not continuous but divided in a discrete set of previously defined concepts. In principle, topics could have a hierarchical relation but there is no natural definition of an order. So the identification of peak bounds as in the temporal dimension ceases to apply here.

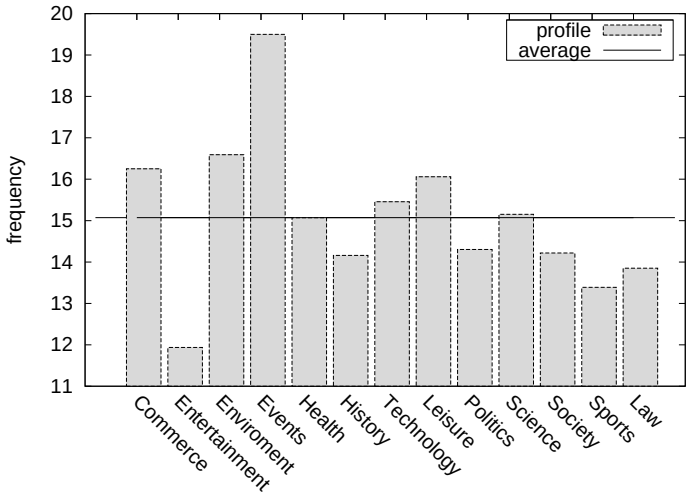


Figure 2.5: Subject Profile of Topic 363: *Transportation Tunnel Disasters*

For the topic classification we need to build abstract models for all different concepts, which the classification should take into account. Language models can be applied as classifiers for this purpose as shown in Section 2.1.1. In order to demonstrate the idea, we used the 12 different topical models from the before presented HARD track experiments (Section 2.3), that correspond roughly to the main sections of common newspapers, like politics or sports.

The required text classification for computing a topical profile differs slightly from the typical categorization task described by Sebastiani (2005). We do not need to assign binary labels whether a document belongs to a certain category or not. A similarity measure showing to which extend an article belongs to a given category is already sufficient. As before we use here the normalized logarithmic likelihood ratio $NLLR(M_i|D)$, where M_i is the language model of the given topical concept. In order to speed up the computation of topical profiles as well as the later ranking procedure the score computation is performed off-line. For each classifier in the set of topical concepts a score vector is maintained, holding the individual scores for all documents within the collection. An example topical profile is displayed in Figure 2.5.

After the classification task is done, topical profiles can be computed in the following way. Similar to temporal profiles explained previously, the set of the 100 top ranked documents given the query is determined. The score for a specific topic concept M_i is then defined by the sum of all document scores from D for this concept. The intensity value, as introduced in the

previous section, is computed accordingly:

$$M_i = \sum_{D_j} NLLR(M_i|D_j), \qquad intensity(M_i) = \frac{M_i}{avg(M)}.$$

2.5.3 The Clarification Interface

After generating and analyzing the query profiles, we discuss in this section how the gained information can be presented to the user for query clarification. The user interface thereby has to fulfill two functions:

- It needs to present all necessary information to the user that allows her/him to take a decision.
- It should provide simple but powerful means to adapt the query in the intended way.

The second point needs further explanation. Not all search topics are easily expressed by a few query terms. Although several articles contain the same keywords, their specific view on the topic or genre might not match the type of documents the user had in mind. If we allow the user to refine the query not only by further keywords but by selecting preferences to more abstract concepts or to restrict the search space to a certain location or time, the difficulty of expressing such context information accurately can be reduced. However, confronting a user in an advanced search interface with all possible combinations of restrictions and preferences to an in general unlimited number of concepts, dates, or locations, would overextend the searcher. Maybe he/she does not even know the correct query meta-data, e.g. the date or location of the event he/she is looking for. Query profiles can help here, since they allow to automatically find the most important meta-data concepts given the initial query terms. This way it is possible to provide the user with the necessary information to set preferences or restrictions and to limit the search dialog to the most interesting options.

Compared to the profiles shown in the last section (Figure 2.4 and Figure 2.5) a user does not need to see the whole spectrum of the profile. Instead it seems sufficient to cut out the most relevant part of it, which means the highest temporal or topical peaks. For the experiments, we just displayed the 5 top ranked topical concepts, but all identified temporal peaks. In practice their number never exceeds 4. In order to demonstrate the usefulness of the profile information and to explain why we restrict the output to the top ranked parts of the profiles, let us distinguish three possible cases:

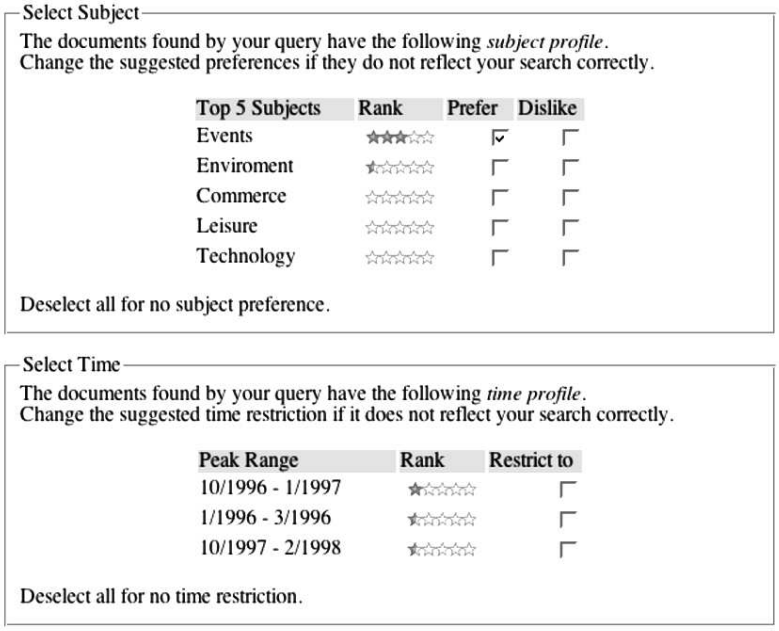


Figure 2.6: Experimental Clarification Form of Topic 363: *Transportation Tunnel Disasters*

(1) In case the initial query was clearly formulated, the user gets a positive confirmation by seeing the expected topic or time partition on top of the ranked profile list, succeeded by close related ones. The absence of non-matching topics will be enough information for the user here. He/she does not need to see a long list of minor ranking topics.

(2) In case the query was ambiguous unwanted topics or time partitions will populate the top of the ranked query profiles. In order to get an unambiguous output, it is now important to refine the query in a way that it excludes most of the unwanted answers, but keeps the relevant ones. Again, the end of the ranked profile list is less interesting, since the topics there are already efficiently excluded by the query.

(3) In case the user does not even find the relevant topics or time partitions among the top part of the query profile, it will not help to just refine the query. Either the query needs to be reformulated entirely or the corpus does not include the documents the user is searching for.

The second case is the most interesting one since it requests appropriate query refinement strategies. Whereas a time restriction based on the profile can be expressed relatively easy, it is in general difficult for a user to find on his

own additional keywords that allow to distinguish between the wanted and unwanted topics of the profiles. However, the system has already abstract classifiers at hand to perform such filtering. The simplest way to refine the query is thus to express preferences directly on the profile itself. For this reason we made our query profiles interactive by adding *prefer* and *dislike* buttons to the topic profiles and *restrict to* fields to the temporal profiles, refining the query in the obvious way. Their exact influence on the final ranking is discussed in the next section.

Automatic Preselection We also looked, whether it is possible to make an automatic suggestion of an appropriate selection in the profiles. Obviously, the most highly ranked topics or temporal peaks are good candidates, especially if they distinctively stand off from the lower ranked ones. The intensity measure defined in the last section explicitly addresses these characteristics. Using an intensity threshold, we can preselect all topics and temporal peaks above. For the later experiments an intensity threshold of 1.2 was used for the topical profiles, respectively 1.5 for the temporal profiles. These values have been shown high enough to assure the selection of only distinctive peaks of the profile. An example clarification form with preselected items is shown in Figure 2.6.

Automatic preselection is especially helpful in the first of the three scenarios above where the query is unambiguous. In such a case user feedback is not necessary and the query refinement could be performed as a sort of "blind feedback" procedure to sharpen the topical or temporal focus.

2.5.4 Score Combination and Normalization

In this section we adapt the previously introduced score combination approach (see Section 2.2.1). The focus lies thereby on the issues of score normalization. When multiple preferences or dislikes have to be handled the logarithmic scores of their corresponding models M_i are simply added, respectively subtracted for disliked models:

$$score(D|M) = \sum_{M_i \in P^+} NLLR(M_i|D) - \sum_{M_i \in P^-} NLLR(M_i|D).$$

The set P^+ denotes all preferred concepts, respectively P^- all disliked.

The final combination of the relevance evidence coming from the initial query $score(D|Q)$ and the meta-query $score(D|M)$ requires further consideration. We have to ensure that the scores on both sides deliver "compatible" values. More precisely, the score of the initial term query should still be dominant in the final result. The introduction of dislike statements might cause

the score of a document to fall below zero. We performed therefore a so-called minimum-maximum normalization, among others described by Croft (2002). It shifts the minimum of a score range $min\_s = \min\{score(D^*)|D^* \in C\}$ to zero and its maximum to 1. We further stressed the importance of the initial query by doubling its score value in the final ranking:

$$\begin{aligned} norm(score(D)) &= \frac{score(D) - min\_s}{max\_s - min\_s}, \\ final\text{-}score(D) &= 2 * norm(score(D|Q)) + norm(score(D|M)). \end{aligned}$$

Since the collection data is date-tagged, the date of a document can be determined without uncertainty. Restrictions on the temporal dimension are treated therefore by binary filtering, removing all documents from the final ranking that do not match the restricted time spans.

2.6 Experiments

Relevance feedback based on query profiles is evaluated in the setting of the HARD track 2005 (Allan, 2004). A set of 50 queries which were regarded as difficult – the query set was taken from the Robust track that tries to tackle selected difficult queries in an ad-hoc retrieval setting – is evaluated on a 2 GB newspaper corpus, the Aquaint corpus. The track set-up allows one-step user interaction with so-called clarification forms that have to fit one screen and have to be filled out in less than 3 minutes. In the original TREC setting the sent-in clarification forms were filled out by the same person who later does the relevance assessments for the specific query. We repeated the experiment ourselves, asking different users to state preferences or restrictions in the clarification forms after reading the query description and query narrative coming with the TREC search topics. This way, we inevitably lose the consistency between clarification and relevance assessment ensured by the HARD setting. However, we could study differences in the user behavior and their results.

The 4 test users – 1 female and 3 male students – were shortly introduced to their task by demonstrating one randomly picked out example clarification form. They needed on average 35 minutes to accomplish the task of clarifying all 50 queries. Most of the time was in fact necessary to study the respective query topic. The preference selection itself was done within seconds. We want to remark here, that the number of test users was rather low. Thus the conducted experiments cannot be regarded as a fully qualified user study, but aim at gathering first indication whether the proposed feedback technique is able to improve retrieval.

	base	*auto*	*user1*	*user2*	*user2**
MAP	0.151	0.187	0.204	0.187	0.201
R-Prec	0.214	0.252	0.268	0.255	0.265
P@10	0.286	0.380	0.396	0.354	0.402

Table 2.2: Result Overview

In order to compare the improvements, we performed a *baseline* run using just the up to 3 words from the query title, further one run with the automatically derived preferences only as explained in Section 2.5.3, referred to as *automatic* run. From the 4 evaluated user runs, we present here the two most different to keep the figures clear. Whereas *user1* selected almost no topic dislikes, *user2* had the highest fraction of dislike statements among his topic preferences. For comparison, we generated the artificial *user2** from the preferences of *user2*, but ignoring all his dislikes.

A closer look at the set of the 50 search topics revealed, that they have not been distinctive with respect to their temporal profile. In fact, there was almost no case where the user wanted to restrict the query to a certain time span. Therefore, we restricted our analysis to the improvements by topical query refinement and ignored the few stated temporal restrictions.

Results Table 2.2 presents an overview on the main evaluation measures computed for all presented runs. At a first glance it is obvious that the refined queries, even in our non-optimal evaluation setting, show a considerable improvement over the baseline run. The precision gain is most visible at the P@10 measures. Since we were mainly aiming at precision gain at the top of the retrieved list, this outcome is quite encouraging. The precision recall graph (Figure 2.7) confirms the observation made with the P@10 values. Also here we observe the highest precision gain at the top of the ranked list. On the right side, the runs with query refinement slowly converge to the baseline, but always stay on top of it. The results of the other two non-displayed users remained always in the middle of the two shown here.

The special run ignoring the topic dislikes of *user2* has a better general performance than its counterpart. Although it is not shown in the table, this observation holds for all four tested users. It indicates that topic dislike statements bear the risk to weaken the result precision in our current implementation.

Surprisingly, the values show also that the automatic run can compete with the user performed clarification. We cannot entirely explain this phenomenon, but can make two remarks on its interpretation. First, the query set has not been designed to test disambiguation. If a query asking for “Java”

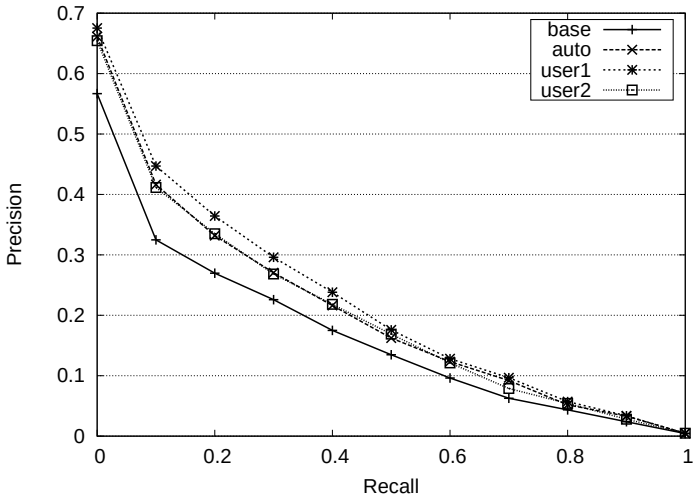


Figure 2.7: Precision Recall Graph

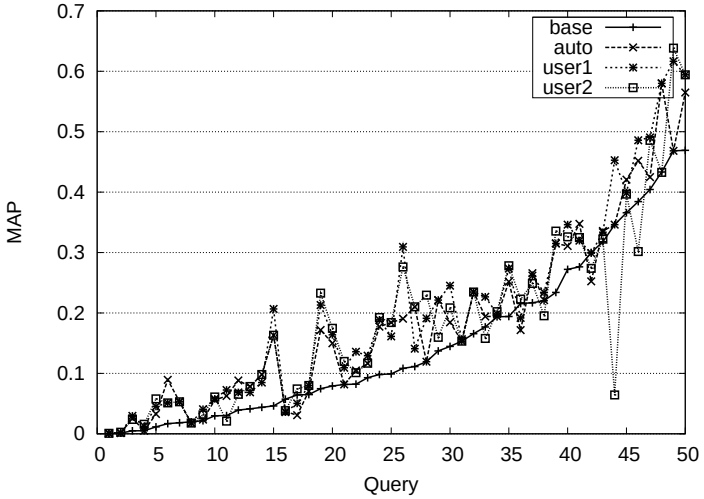


Figure 2.8: MAP Improvements on Single Queries

expects documents about the programming language, automatic topic feedback will work perfectly. However, it fails if in fact the island was meant. Examples of the second type are necessary to compare user and automatic feedback, but are not found in the test set. A further reason for the good performance of the automatic run might simply be the fact that it does not contain dislike statements.

For a more detailed view on the results, Figure 2.8 presents the evaluation of all single queries sorted by increasing MAP value of the baseline run. Thus, the graphic shows the worst performing queries on the left, continued by a section with still relatively low quality response in the middle, up to acceptable or even good queries on the right. Although the improvement per query is not stable, it seldom happens that the user feedback deteriorates the results. The one extreme case on the right side of the figure is again caused by dislike statements. If we consider the relative improvement, the queries in the middle part of the figure apparently gain the most from query refinement. Within the distinction of query types from Section 2.5.3 these queries probably fall under the ambiguous category 2. The fact that we encounter the highest improvement in this area nicely demonstrates the usefulness of our method.

2.7 Summary and Conclusions

This chapter was aiming at retrieval refinement by making use of features of the query context. We first discussed an appropriate modeling of the query context, with the conclusion that user-independent concept models come with considerable advantages compared to direct models of the user. Furthermore, we pointed out that language models are suitable as a uniform representation of concept models and allow to build a uniform approach for context scoring.

We also developed a framework for score combination that allows to combine individual relevance estimations of all involved concept models as well as the relevance to the initial term query. Initial tests on the HARD track query set and collection showed indeed clear improvements using our context modeling and combined ranking approach. However, the score combination was not yet capable of handling "dislike" statements appropriately, that result from user feedback on the query profiles. Further analysis is needed of how to make use of topical dislike statements in a way that they do not harm the results, but also contribute to the query refinement. Incorporating negative user feedback correctly is in fact a known problem in information retrieval (Ruthven and Lalmas, 2003). Moreover, both experimental evalua-

tions on the HARD track data of 2004 and 2005 had to limit the considered context dimensions, since no concept models were available or the query set was not sensible to some of the context dimensions. Hence, the experiments never used a larger set of query dimensions and cannot show how the score combination approach would work in scenario of rich query meta-data.

When using query meta-data for improving the retrieval precision, we had to explain where such meta-data might be taken from. We developed therefore a new type of feedback approach, that gathers query meta-data in an interactive retrieval session without asking the user unnecessary questions. The proposed approach employs so-called query profiles and has been introduced in comparison to other existing feedback methods. We also explained how query profiles can be computed and analyzed for exceptional peaks, that play an important role in query refinement.

The results show promising improvements for all runs that make use of query profiles even in our preliminary experimental study. With a query set designed to test how retrieval systems cope with ambiguity, we would probably be able to show even higher improvements using our feedback method. The same applies for queries that reward temporal restrictions. The lacking of a testset with more ambigious queries and corresponding relevance assessments is a known problem (Spärck-Jones et al., 2007).

A finer grained topical "resolution" potentially in form of a topic hierarchy, could lead to more focused query profiles on the topic dimension. Furthermore, we need to examine query profiles on other context dimensions. The temporal profiles remained untested by the current HARD track query set, but also geographical or genre profiles - to name just two possible other parameters - might enable similar improvements as the topical query refinement.

The automatic feedback method turned out to be an interesting side product of the work with query profiles. It performed almost as good as the user feedback. It raises the question to which extend the system can decide based on query profile statistics, whether automatic feedback is reliable enough in a certain case to omit user interaction. Especially when several context dimensions are involved in the analysis, the user should not be delayed by a multiple number of feedback questions. Instead an intelligent retrieval system might be able to select the most helpful dimension for explicit user feedback itself.

3

Structured Retrieval on XML

Whereas most common retrieval models regard a document as a simple "bag of words", humans see far more when they look at a document. Even a short glimpse is sometimes enough to judge a document as irrelevant without even reading the content properly. Such fast judgment is mainly helped by the structure of the document. Layout, titles, paragraph-lengths and many more features contain valuable information about the genre and content of the document. Still, such features are completely neglected in the bag-of-word model, and therefore cannot be exploited by most IR systems.

So-called mark-up languages such as SGML or XML are widely used nowadays to annotate text structure in a machine readable form. It is hence a straightforward aim to exploit such mark-up for retrieval. Research in the field of structured retrieval has therefore focused on working with XML data in the last years, mostly driven by the INEX evaluation initiative (Fuhr et al., 2005). This chapter will touch several aspects of XML retrieval research. In particular, how to design a query language for addressing structural constraints and how to perform efficient evaluation of typical structured retrieval tasks. The presented research stays in close connection to the development of our open source XML retrieval system *PF/Tijah* (Hiemstra et al., 2006), which is integrated as a module in the XQuery compiler *Pathfinder* (Boncz et al., 2005) and executed on the main-memory database back-end *MonetDB* (Boncz, 2002). The introductory section on structured query languages is partly based on previous work (Hiemstra et al., 2006).

3.1 Query Languages for Structured Retrieval

Whereas users in general know how to express content-related queries by keywords they do not know how to express structural constraints. Structure

is easy to recognize, but often hard to describe. Structure is also never the primary aim of a search. Instead, queries on content are refined by structural constraints. Pure structural search is in fact more common in the domain of "database queries", for instance with the aim to filter out a certain type of elements. However, such queries do not ask for a ranking of the results.

3.1.1 Structural Features of XML

When designing query languages for the expression of structural search properties, there are two main points to consider. On the one side, it is important to analyze the structure of the XML data, on the other side, we have to take into account what structural features are helpful and desirable for search.

We will start here by looking at the actual structure of common XML collections. XML data is typically categorized as either data-centric or document-centric (Fuhr and Groβjohann, 2004). The distinction relates to the homogeneity of the structural annotation and to the content of each element. An address-book with all its optional fields for each entry would be a typical instance of data-centric XML. The XML tagging is used here to split data entries into their respective fields. The structure is highly homogenous. Even with optional address fields, it is usually defined which elements are allowed in a given context. Furthermore, the content of such fields is typically short and of a certain type, e.g. postal code, date, or names. In contrast, the often seen XML version of the plays of Shakespeare can be regarded as typical document-centric mark-up. The structural tagging is used here to divide a large amount of text into a hierarchy of meaningful units, being an act, a scene or just the phrase of a specific speaker. We can observe for both types of XML that the hierarchical order of tags is more consistent than the arrangement of tags in reading order, so-called document order in XML. If a DTD or XML schema is available, the patterns most often describe parent/child relations between two tags, e.g. `<TD>` being allowed only inside `<TR>`. In contrast, the sibling order of elements is rarely defined in a DTD.

In the context of text search, the document-centric type of XML markup will play a dominant role. Data-centric XML is queried as well, but more by database-style selections rather than by vague ranking criteria as used in information retrieval (Fuhr and Groβjohann, 2004). The typically short content of elements in data-centric XML is less suitable for queries that involve the ranking of results. Since XML emphasizes the hierarchical structure more than the document order, we will also find querying hierarchical properties more important than document order features.

Looking more from a user perspective, a major problem of structured XML retrieval is the lack of standardization in the usage of structural an-

notation, especially in heterogenous collections. Different tag names might be used to mark equivalent units of text, e.g. `<HEADLINE>` vs. `<TITLE>`, and even the existence of a structural mark-up cannot taken as guaranteed for all elements. XML retrieval systems, in consequence, often interpret structural constraints vaguely, regarding them as hints rather than as requirements. They try to prevent this way the unwanted exclusion of possible relevant answers that had been annotated differently (Fuhr and Groβjohann, 2004).

Kamps et al. (2006) categorized structural queries as used in the INEX evaluation initiative in order to analyze what kind of structural properties users want to express in their queries. The underlying assumption of the study is that INEX queries are typical for structured user queries. This is in fact questionable, since the queries had to be expressed within in the limits of a certain query language (NEXI) and they are written by researchers with the primary aim to evaluate their systems. Nevertheless, the outcome and classification of the queries is of interest here. Queries are divided in using so-called *hierarchical* and/or *context* properties. Note that the notion of hierarchical queries differs from how it was used in this section before. The requested output elements are taken as a reference point for the distinction. Hierarchical features refer to descendent elements of the final output element, like searching for *sections* on "*XML retrieval*" having *subsections* about "*databases*". In contrast, context features describe the neighboring elements, not included in the output itself, e.g. used when looking for *sections* on "*XML retrieval*" in *articles* with "*databases*" standing in the *title*. Interestingly, the later category of queries was found clearly more often within the INEX queries. This observation is encouraging for structured retrieval, since those queries differ entirely from simple fielded queries and require more flexibility from the query language and the retrieval system.

3.1.2 General Query Language Requirements

In order to enable the formulation of contextual (and also hierarchical) queries, the desired query language should provide a certain set of functions on XML element nodes. Mihajlovic (2006, Chapter 3.1) presents a minimal list of functional requirements for structured retrieval:

element selection: Selecting element nodes of a specified tag-name, or the set of elements of different given tag-names.

element scoring: Scoring any node set by the estimated relevance to a given text query.

containment evaluation: Given two node sets of ancestor, respectively descendant candidates, evaluate which node pairs fulfill the containment condition. The evaluation also needs to propagate existing scores towards contained/containing nodes.

score combination: Combine different scores of the same nodes.

We renamed the score propagation function mentioned by Mihajlovic in order to stress the containment evaluation aspect. Although the attendant score propagation plays an important role, we regard it more as a side effect of the containment evaluation looking from the perspective of a query language.

The first two requirements allows simple fielded search (see Section 1.1), e.g. document retrieval on title words only. Especially for heterogenous collections with changing structure, it is important that the user is free to rank elements of any given tag-name. The containment evaluation further enables to ask for the hierarchical relation of nodes. We argued above (see 3.1.1) why hierarchical features are more important in XML retrieval than document order features. For a minimal list of requirements the containment relation seems sufficient. It allows to express conditions on contained elements as well as on containing ones. The score propagation and combination together enable to combine the rankings of contained and containing nodes.

In the following, we examine two different structured query languages that have gained attention in the research community. The languages are shortly introduced and compared to the above requirements.

3.1.3 NEXI

The NEXI query language (Narrowed Extended XPath I: Trotman and Sigurbjörnsson 2004) was designed with the needs of the INEX community in mind. It should remain as simple as possible – for users as well as for system developers, not bound to a specific approach, but at the same time capable to experiment with querying content and structure. The syntax is based on the navigational XPath language being a W3C recommendation.[1]

O'Keefe and Trotman (2003) explain why and in which way XPath was restricted and extended to better meet the needs of the INEX community. One of the main changes concerns NEXI's restriction of the navigational axis steps. NEXI knows only two of the 13 XPath axes, namely the `descendant` and `attribute` axes, in case of the `attribute` axis even with slightly different semantics. The restriction was introduced after observing a high number of incorrectly formulated queries leading to unexpected empty results. Due

[1] `http://www.w3.org/TR/xpath`

to misconceptions of the document structure, users were for instance asking for child steps where the wanted element nodes stood in fact only in ancestor/descendant relation. The language restriction thus simply avoids common errors that are even common with expert users, as represented by the group of INEX researchers. Notice that NEXI still satisfies the claim for containment evaluation in the above listed requirements for structured query languages.

NEXI also extends XPath to enable querying the element content. A special `about`-function is introduced to filter and rank a set of element nodes according to their text content. As an example, consider a query looking for *paragraphs* about *XQuery* in *html* documents about *information retrieval* and *databases*. The corresponding NEXI query would look like the following, assuming the evaluation to start at the collection root:

```
//html[about(., ir db)]//p[about(., xquery)]
```

Although the `about`-functions are used here inside predicates evaluating to boolean type, the NEXI semantics require implicit score propagation and combination. Hence, the final ranked list is influenced here by both document and the paragraph scores. With the `about`-syntax and the implicit score combination and propagation NEXI also fulfills the other requirements of structured query languages.

Furthermore, NEXI introduces a shorthand writing to express element name-filtering on a set of tagnames, like `//html//(title|headline)` in place of the longer: `(//html//title|//html//headline)`. The abbreviated syntax is meanwhile also allowed in the XPath standard. It addresses the mentioned problem of heterogenous collections, where different tagnames were used as mark-up for semantically equivalent structure. In conclusion, NEXI indeed meets its design goal of addressing the minimal needs of a query language for both structure and content, without introducing expressional power that might cause unaware misuse.

3.1.4 XQuery Full Text

XQuery is a functional database query language developed by the W3C to become the standard for querying XML data, much like SQL is for relational data.[2] XQuery comes with a clear data-centric view of XML. It combines powerful selection expressions on existing data with the possibility to compose the results in any XML format by creating arbitrary new elements. However, XQuery itself does not have any text retrieval features to support

[2] `http://www.w3.org/TR/xquery`

IR-style ranking queries. To overcome this shortcoming, XQuery FT (full text) is designed as an extension to XQuery introducing full text search functionality into the query language (Amer-Yahia et al., 2007). As a language extension, the XQuery FT expressions have to satisfy several additional requirements, like being side effect free, fully composable with XQuery and using the same data model. Rys (2003) lists all those requirements and explains the decisions of the language design.

The above introduced example query would be expressed in XQuery FT like:

```
let $c := doc("mydata.xml")
for $res score $s in
$c//html[. ftcontains ("ir","db")]//p[. ftcontains "xquery"]
order by $s descending
return $res
```

Unlike NEXI, XQuery returns result sequences usually in document order. Therefore results have to be ordered by score explicitly to achieve a ranked list output. The special `score` syntax in the `for`-loop binds the scores of the corresponding expression to a variable, which can later be used to express for instance a score threshold or an ordering on scores like in the example. The syntax extension of the language became necessary, since score expressions are inherently second order functions, taking another expression as their argument (Rys, 2003).

XQuery FT also gives the users by far more expressive power than NEXI. In contrast to the semantically "safe" restriction to the `descendant` axis, it allows to use all XPath axes. The language thus assumes an expert user, who knows the structure of the queried data. Moreover, XQuery FT implicitly performs score propagations among all axis steps and combines scores of different subexpressions. It satisfies thus all four requirements for structural query languages, however, especially those implementation defined implicit score propagations and combinations make it difficult to design a sound scoring framework. A user would for instance expect the following query to be semantically equivalent to the above shown[3]:

```
let $c := doc("mydata.xml")
```

[3]The equivalence results here from:

```
    //a[. ftcontains x]//b[. ftcontains y]
⇔   //b[ancestor::a ftcontains x][. ftcontains y]
⇔   //b[ancestor::a ftcontains x and . ftcontains y]
```

```
for $res score $s in
$c//p[./ancestor::html ftcontains ("ir" && "db")
      and . ftcontains "xquery"]
order by $s descending
return $res
```

Since the retrieval model and score propagation are not defined by the query language but left to the implementation of the system, it is also in the responsibility of the retrieval system to take care of such semantic equivalences. In fact, we found that many common retrieval models and propagation approaches do not return an equivalent scoring in the above case.

XQuery FT comes furthermore with a set of additional functions for explicitly performing proximity, thesaurus, or wildcard queries and to express further retrieval options like stemming. Those functions challenge the performance of retrieval systems that rely on the existence of pre-computed in index structures. An index build on a stemmed term vocabulary, will not be able to answer queries that explicitly asks for the use of unstemmed forms. Hence, retrieval indices will have to be highly redundant to fulfill all possibilities of XQueryFT. For the work presented in this thesis, the support of these additional language features is lying out of the scope, though the later proposed index structure is able to deal with a number of them.

3.1.5 NEXI Embedding in XQuery

When Rys (2003) explains the integration of text retrieval features in the XQuery language, he distinguishes three principally different possibilities, namely a (1) sublanguage, a (2) functional, or a (3) syntactical approach. A functional approach does not need any language adaptation, but introduces a large set of highly parameterized functions for each required text search feature, resulting in long and unreadable queries. Also the sublanguage approach was undesirable for the design of XQuery FT. It integrates an independent sublanguage for querying and scoring via a minimal set of functions into the existing XQuery language. The embedded sublanguage query, however, remains here a simple "black box" string inside XQuery, which restrains its parameterization and compositionality with XQuery. The syntactical approach differs from the other two in that it requires an extension of the query language with new keywords and grammar rules, but provides the most flexible and expressive integration. XQuery FT has chosen the last option with the introduction of the `score` construct and the `ftcontains` expression with all its optional syntax for e.g. stemming or proximity.

Despite the mentioned disadvantages, we chose the sublanguage approach when designing the first query language for our own research search system

PF/Tijah (Hiemstra et al., 2006). It integrates NEXI as a text search sublanguage into a standard XQuery system. In the following, the advantages and problems of the language embedding will be shown and discussed.

Starting with an example, the above presented search task is expressed in PF/Tijah by the following query:

```
let $c := doc("mydata.xml")
return tijah:query($c,
   "//html[about(., ir db)]//p[about(., xquery)]")
```

The newly defined function `tijah:query` takes as its arguments a sequence of so-called *start nodes*, often the document root, and a NEXI query string, returning a resulting node sequence in decreasing order of relevance. The query evaluation is rooted at the start nodes sequence. The embedded NEXI expressions become compositional this way with the surrounding XQuery.

The sublanguage embedding brings together the strong aspects of both XQuery and NEXI. The PF/Tijah approach is able to combine in one query the expressive power of XQuery for selections on data-centric XML with text search features formulated within the semantical "safe" restrictions of NEXI. We can for instance filter the resulting ranked node sequence by an XQuery expression selecting those paragraphs written by a given author. Unlike NEXI, XQuery also allows the user to specify the output presentation of a query by generating arbitrary new XML elements. It is simple to create for instance a list of author and title elements instead of the corresponding ranked articles. Finally, the self-defined `tijah:query` function already returns a sequence in descending ranked order, which is in most cases handier than the output of the XQuery FT functions sorted by document order.

As already mentioned, the sublanguage approach also comes with disadvantages. The following complex example query demonstrates several difficulties. Consider running a TREC style evaluation, which executes 50 queries found in a separate topics file. The following simplified code shows a solution that performs the entire evaluation at once:

```
let $c := doc("mydata.xml")
for $q in doc("topics.xml")//top
  let $num := $q/num/text()
  let $query := concat("//DOC[about(.,", $q/title/text(), ")];")
  let $id := tijah:query-id($c, $query)
  for $doc at $rank in tijah:nodes($id)
    where $rank < 1000
    return string-join(
    ($num, $doc/DOCNO/text(), $rank, tijah:score($id, $doc)), " ")
```

Without the syntactical `score` construct of XQuery FT, a query identifier is necessary here as an indirection to return later the nodes as well as the corresponding scores as a second order aspect of the sublanguage query. Three function calls are required in this case for one single query. Another difficulty concerns the query parameterization. Since the NEXI query remains a black box string for the XQuery system, the query can only be modified by less elegant string concatenation as done in line 4 of the example above.

If we look more from the systems point of view, the sublanguage approach comes with the disadvantage that it does not allow static code checking or query compilation. The interpretation of the sublanguage query can only be done at runtime, when the actual query string is evaluated. However, the self-contained sub-queries allow a system design, where the NEXI subsystem and the surrounding XQuery engine remain independent to a large extend. For our own research system PF/Tijah this last point became the major decisive factor for the sublanguage approach, since it simplified the integration of two existing predecessor systems.

3.2 Indexing XML Structure and Content

After discussing query languages that enable the user to search collections with respect to structure and content, we can now proceed by addressing the question how to evaluate such queries efficiently. Once the typical query execution patterns are known, we can study their data access and try to enhance the data access by building indices. In fact, the actual data access highly depends on query plans and employed low level algorithms, both discussed in section 3.3. Still it seems appropriate to first introduce indexing techniques in general, and the *PF/Tijah* index in particular.

3.2.1 Data Access Patterns

Structured queries, as expressable in the introduced languages, can be supported for efficient evaluation by indices. However, in order to create the appropriate indices, we need to know the data access patterns of such queries. The simplest – and probably most often occurring – structure and content query asks element nodes of a certain type (e_1) with text content on a keyword ($t_1 \dots t_n$) defined topic. In NEXI syntax:

```
//e1[about(., t1 ... tn)].
```

In order to evaluate such a query on a given text corpus, the retrieval systems needs to

(1) find all element nodes with the specified tagname (e_1) and all term occurrences of the keywords ($t_1 \ldots t_n$),

(2) evaluate the containment of term occurrences and element nodes, thus finding all tuples (`e`,`t`) of a keyword occurrence `t` in the extent of an element node `e`,

(3) access further scoring model dependent data: e.g. element sizes, or collection-wide term-counts. Retrieval models often compare local (document specific) and global (collection specific) probability distributions of term occurrences. The global statistics are usually pre-computed and have to be accessed as well.

In place of the simple fielded search above, we will also find more complex structural expressions: like

```
//e1//e2[about(., t1 t2)]//e3[about(., t3)].
```

However, with respect to the necessary data access, the complex query does not show new access patterns. The rooted path expression `//e1//e2` and the trailing `e3[about(., t3)]` only require to evaluate the containment relation of element nodes. In all cases, containment is interpreted here as following the descendant axis, thus including children as well as indirect descendants. This applies also for text nodes being contained in all their ancestors.

3.2.2 Indices for Content and/or Structure

The following short overview on content and structure indices will introduce and examine existing indices and says whether they support the above listed operations.

Inverted Document Indices Retrieval systems most commonly make use of inverted document indices to efficiently access all occurrences of a given term in a collection.

Such indices maintain a posting list per term containing all document identifiers of documents that mention the term. Several techniques have been developed to further improve the index performance, e.g. by compression techniques or pruning of probably unimportant postings. Zobel and Moffat (2006) give a good overview on the issues around inverted document indices.

term	postings
⋮	
information	2, 10, 23, 117, 118
retrieval	23, 64
⋮	

Figure 3.1: Inverted Document Index

In the context of XML, however, the well-proven index structure fails due to the changing notion of documents. All element nodes can theoretically be regarded as documents. Hence documents might be highly nested. In consequence, a posting list would not contain exactly one but multiple entries for each occurrence of the given term for all surrounding element nodes. The overall index size would grow roughly by the factor of the average tree depth, and thus become highly inefficient. The problem can be solved partly by listing only the direct parent element of a term occurrence in the inverted index and not all its ancestors. This way, each term occurrence is again mentioned exactly once. Still, such an inverted parent index comes with major differences compared to the conventional document index. Asking for term occurrences within an arbitrary set of elements, the listed parent nodes are not the final answer. An additional structural containment join is necessary to decide which of the indexed nodes are contained in this set of elements.

Relational Tree Encodings The immediately following question is, how index structures can support the containment join between two sets of nodes. The structure of XML documents can be represented by a tree, where element nodes are mapped to vertices and the parent-child relation of two nodes is shown by directed edges between the corresponding vertices. The containment of two nodes can be evaluated by searching for a path in the tree between the two nodes. However, neither can the entire tree be held in memory for large XML documents, nor is it possible to efficiently check the indirect containment relation between two nodes.

Relational tree encodings have been designed in the database community to tackle the problem. On the one hand, *Dewey*-based encodings assign labels to each node that capture the complete rooted path to the node, similar to the typical section/subsection numbering of larger documents (O'Neil et al., 2004; Tatarinov et al., 2002). On the other hand, region encodings simply enumerate the XML nodes in document order of their start and end tags, assigning so-called *pre* and *post-order* values (Grust et al., 2004; Li and Moon, 2001; Zhang et al., 2001).

```
<a>
   <b/>
   <c>
      <d/>
      <e/>
   </c>
</a>
```

pre	post	tag
1	5	a
2	1	b
3	4	c
4	2	d
5	3	e

Figure 3.2: Document and Pre-Post Index

Those two values are enough to perform containment checks between a pair of nodes x, y:

$$x \text{ contains } y \equiv pre(x) < pre(y) \wedge post(x) > post(y).$$

Dewey-based encodings are theoretically better in handling updates of the index than region encodings – an index property not discussed here so far. Changes only influence the local numbering of the corresponding subtree, whereas in the region encoding all globally following nodes needs to get assigned new numbers. The main disadvantage of Dewey encodings, however, is the length of the assigned labels. Notice that they require to maintain a number for each level of tree depth of a node, whereas region encodings just need integer values. Especially when handling containment joins of large node sets, the simpler integer comparisons are performed more efficiently.

Zhang et al. (2001) showed that it is possible to efficiently store XML data and to process structural queries by the combination of an inverted text node index together with a region index. The later presented *PF/Tijah* index in fact most resembles this approach.

Further Indices for Content and/or Structure So-called *DataGuides* summarize the hierarchical structure of the XML tree (Goldman and Widom, 1997). The index tree contains all distinct labeled rooted paths. Each index node describes thus a distinct class of element nodes. A complete rooted path query is then evaluated first on the considerably smaller index tree. In a second step, all instances of the query satisfying index nodes are fetched to be returned as the final answer. Notice that DataGuides show their biggest advantage, when evaluating path queries consisting of long rows of child steps, e.g. `/a/b/a/c`. Such queries lead to exactly one qualifying index node and all instances can be fetched directly without overhead. Whenever the path query contains descendant steps – the only allowed axis step in NEXI – the evaluation on the index tree yields multiple possible answers and requires to fetch the instances of several index nodes. However, when the path query contains predicates, it is impossible to evaluate the query based purely on such a DataGuide index.

Ramírez and de Vries (2004) suggests to maintain a DataGuide in addition to the relational tree encoding. A query optimizer can then decide by simple heuristics to use the DataGuides for appropriate parts of the query. Weigel et al. (2004) and Kaushik et al. (2004) designed a more content-aware DataGuide[4]. They couple the inverted lists tighter to the DataGuide by maintaining for each term posting also the DataGuide node corresponding to the direct ancestor element of the term occurrence. When term postings are

[4]Kaushik et al. (2004) actually uses a different terminology. DataGuides are just called structure indices here.

fetched from the inverted list, it is then possible to select only those occurring under a specific rooted path, which saves the loading of non-matching term occurrences. The later query evaluation strategy differs slightly, but both show how even branching path expressions including predicates can make best use of the DataGuide. The disadvantages of the approach are lying in the more complex index structure and, more important, in the fact that they support direct containment queries better than indirect ones. When several DataGuide nodes are matched by the path query, the inverted lists of all these nodes have to be loaded and merged causing new evaluation overhead. Furthermore, it is in such case impossible to return the matching tuples of elements and contained term occurrences without performing at least one final containment join on the instances of those two lists. Recall that indirect (term) containment is by far more seen than direct containment in structural queries.

3.2.3 The PF/Tijah Index

The *PF/Tijah* index is designed to support the evaluation of NEXI queries but at the same time remaining a simple and space efficient data structure. It should be possible to entirely (re-)build the full-text index of a collection of text documents as fast as possible. With respect to query evaluation, we consider the support of often occurring query patterns, like simple fielded queries, more important than specialized complex path queries.

Having these design goals in mind, we abandoned the use of DataGuides. Instead a simple region encoding combined with inverted indices to look up term and tag occurrences is employed. In relational terms, we store for each element node e the tuple:

$$< tag(e), pre(e), size(e) > .$$

And similarly for each term occurrence t:

$$< term(t), pre(t) > .$$

The *pre/size* encoding is equivalent to the before mentioned *pre/post* representation. The value $size(e)$ denotes the number of descendants of element e. In contrast to the XQuery data model (Fernández et al., 2007), we tokenize the character content of an element node into separate terms represented by adjacent text nodes. Those text nodes are also assigned their own *pre-order* identifier, as shown in Figure 3.3.

The advantage of this numbering is twofold. Firstly, the region encoding allows the employment of efficient structural join algorithms to evaluate the

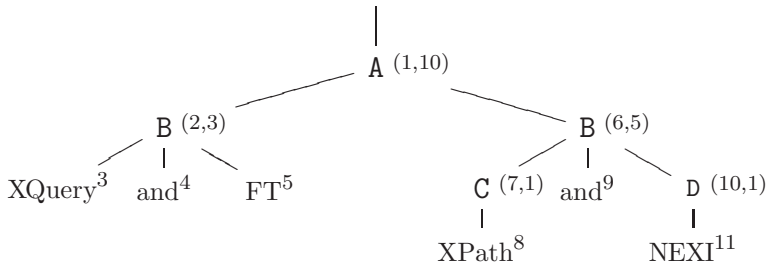


Figure 3.3: XML Tree with word enumerating *pre*/*size* encoding

containment of term occurrences. Secondly, the element sizes maintained in the table estimate the number of terms found in their extend. This number represents a feature of many scoring models. Using the element size instead of the exact term counts assumes that the number of descendant elements is neglectable in comparison to the number of terms. In fact, we have seen in pre-tests that the retrieval quality is not suffering from the overestimation. Furthermore, the enumeration of words enables to ask for phrase or proximity features, an additional useful feature not discussed so far, since it is not bound to structured queries. The *pre-order* values are used in that case as a positional index, showing the exact order of terms.

Using MonetDB The *PF/Tijah* system is operating on the main-memory database back-end called *MonetDB* (Boncz, 2002). The backend comes with a couple of features, that further influence the actual index design. Most importantly, it requires full vertical fragmentation of relational tables, so that – at least on the physical level – all maintained tables are two-column BATs (Binary Association Tables). Whenever one column of a BAT stores a dense ordered sequence of unique identifiers, it gets the special data type `void`. Instead of maintaining all identifiers, only the offset is kept and the two-column BAT is stored in a one-dimensional array. Boncz (2002) describes these system features in more detail.

In case of the inverted term/tag index, the physical concepts of the DBMS changes the index in the following way. Figure 3.4 visualizes the physical storage schema of the relation $<term(t), pre(t)>$. The exactly same structure is used again to maintain the tuples $<tag(e), pre(e)>$. The missing column $size(e)$ has to be stored in separate BAT. Instead of repeating the string representation for all term/tag occurrences, the first of the three presented BATs serves as a dictionary table, that assigns identifiers to all unique terms, respectively tag-names. The actual inverted lists are maintained in two `void`-column BATs, where the second (the most right in the figure) holds all term

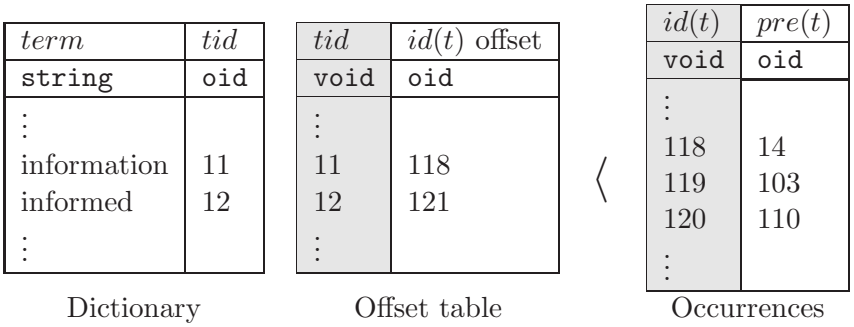


Figure 3.4: Inverted Index in MonetDB, gray-shaded columns are not stored physically

pre(*e*/*t*)	*size*(*e*/*t*)
void	int
1	213
2	0
3	0

(a) Index schema I

id(*e*)	*size*(*e*)
void	int
1	213
2	164
3	420

(b) Index schema II

Figure 3.5: Two variants for the physical storage of size information

occurrences sorted by term, and the first keeps for each term an offset pointing to the first term occurrence of a given term. The *pre-order* positions of any term can be sliced out of the second array after looking up the corresponding offsets in the first table.

Two variants have been tested to store the *size* information (see Figure 3.5). The first possibility (index schema I) is to maintain a *pre/size* BAT with `void`-column *pre-order* keys. Such BAT allows direct positional access to the *size* value of any given node *pre*(v). Therefore, it supports containment joins of arbitrary node sets. However, it comes with the disadvantage of also maintaining unnecessary *size* values of all term nodes, in order to keep the *pre-order*-column dense numbered. An alternative that still allows direct positional access to the node sizes, but does not require the redundant storage of all zero-sized text nodes, is to maintain a `void`-column *size* BAT aligned to the inverted tag index. This option will be referred to as index schema II. Whenever the node positions are fetched of a given tag name, the according sizes can be attached from the aligned *size* BAT. Notice, that the *pre-order* values are not used any longer as the key to access the sizes. Instead, the enumeration in the occurrence array (the rightmost BAT in Figure 3.4) de-

	dict. size	#elements	#terms	XML size	schema I	schema II
Shakespeare	17×10^3	179×10^3	395×10^3	7.9 MB	4.7 MB	3.2 MB
Aquaint	0.8×10^6	21×10^6	253×10^6	3.0 GB	2.1 GB	1.1 GB
Wikipedia	2.2×10^6	53×10^6	151×10^6	4.5 GB	1.5 GB	1.0 GB

Table 3.1: Overview on Index Sizes

fines an element's identity $id(e)$. Due to the alignment with the occurrence BAT, the node sizes are grouped by their corresponding tag-name. Apart from the smaller index size, the grouped storage of node sizes utilizes the systems cache lines better, since most queries score all nodes of a certain tag name. However, as we will see in the next section, efficient containment join algorithms require *pre-order*-sortedness of their operands. It is possible to keep also in index schema II the *pre-order*-sortedness within each tag-name group. The disadvantage of the tag-aligned *size* index, lies in its inflexibility, when queries do not score all elements of a single tag-name. Queries asking to score all element nodes independent of their name, or queries that require to score arbitrary sets of nodes, cause random jumps during the containment join evaluation.

3.2.4 Experiments

The experiments in this section demonstrate the effects of the index design on the space and performance dimension. We will especially compare the two proposals (index schema I and II) for maintaining the sizes of element nodes.

Index Size The *PF/Tijah* index compresses the information hold in an XML document that is necessary to answer structure and content queries. We will report here the actual index sizes for 3 selected example collections: (1) the XML collection of all plays of Shakespeare[5], (2) the Aquaint newspaper corpus[6] used in several TREC tracks, (3) and the INEX wikipedia collection. Whereas (1) is a small, very homogenous collection with the vocabulary of just one author, the other two represent larger collections consisting of meta-data enriched news data (2), respectively structured hyperlinked encyclopedia texts (3).

Table 3.1 shows the number of unique terms after porter-stemming, the number of element/term occurrences in the collection, and the actual sizes of

[5]downloaded from `http://www.ibiblio.org/bosak`.

[6]the original SGML version is converted to XML with the least possible changes.

the original XML files vs. the index files. Schema I and II refer to the entire index sizes including the *size* BAT as described in two proposals. It must be remarked, that the reported index sizes still remain in the same range as the sizes of the original XML text files. The large index sizes are mainly caused by storing all entries in arrays of fixed-length 4 byte integers. Further light-weight compression techniques would obviously be able to decrease the index sizes, but have not been employed so far.

Performance In order to test the performance of the two index schemes in a realistic scenario, a text collection was indexed, and the execution times where measured while running a set of keyword queries of the form:

```
//e[about(., t1 ... tn)].
```

The simple query form includes already all operations involved in the evaluation of structured queries. More complex queries are typically composed of patterns of this simple type. The used text collection in this case was the Aquaint corpus (see Table 3.1). As a set of typical keyword queries, TREC's 2005 robust track queries have been chosen. The query set consists of 50 topics, described first by a few keywords, the so-called query title, and later in a more verbose query narrative. We will analyse in this chapter both the performance of short title-only queries as well as long queries created by the concatenation of title and narrative field. After stop word removal the set of short queries contains 2.5 words on average, respectively 27.4 words in case of the long queries. In contrast to the original queries that always ask for a document ranking, we varied the requested scored element set among the 50 queries, asking alternately to score paragraphs, documents, the documents text body, or the documents title. This way, we tried to create a realistic scenario to measure the performance of different query plans for scoring arbitrary element sets. For the actual score computation, the *NLLR* retrieval model (see Section 2.2.1) was employed which represents a typical log-based scoring method following the language modeling approach. The test system used for all time measurements in this thesis was an AMD Opteron 64 bit machine running on 2.0 Ghz with 16 GB main memory. The index structures could hence be hold in memory, but not in the considerably smaller CPU caches. The entire set of 50 queries were evaluated 5 times and we report the fastest out of 5 runs. The observed deviation between the 5 runs varies only minimal in all observed cases.

Table 3.2 shows the execution times of running the entire query set using either index schema I or II. The evaluation used query plan P2, which will be explained in Section 3.3.3. Apparently, index schema II does not only

	short queries	long queries
schema I	17.6 s	80.2 s
schema II	7.5 s	66.5 s

Table 3.2: Overview on Index Performance

show less space requirements but also a clearly better query performance in general. The difference between the two indexing schemes is higher for the short queries than for the long ones. When executing the queries, the elements have to be accessed once independent of the size of the query. The higher number of query terms only increases the probability that elements contain at least one of the terms which requires their size values to be fetched during the evaluation.

3.3 Scoring XML Elements

After introduction of the index structure, this section will go into detail with the efficient scoring of XML elements. Assuming that a set of element nodes E is selected – E resembles the document set in common retrieval models – and a query Q is given as a set of terms, the scoring of XML elements can be described as a 4 step procedure:

(1) selection of query term occurrences $T = \{t|term(t) \in Q\}$ in the collection,

(2) containment join $E \bowtie T$ resulting in tuples (e, t) where e contains t,

(3) calculation of the score contribution per tuple (e, t),

(4) score aggregation per element.

Step (1) does not need further discussion since it represents a standard database operation supported by the introduced index structure. Calculating the score contribution of each element-term tuple (3) depends on the employed retrieval model. Apart from fetching element and term statistics, this step involves only standard arithmetic functions. Step (4) represents a typical aggregation operation. Its execution can become rather costly when the query size increases. Techniques have been developed in the field of document retrieval to optimize the aggregation costs in such cases (see e.g. Anh and Moffat, 2006). Most specific for XML retrieval and influential on the overall performance remains the execution of the containment join (2), which is studied in the following. Notice, furthermore, that the outlined scoring procedure is not entirely fixed with respect to the order of steps. We

will see later (Section 3.3.3) that different query plans can be considered for their execution.

3.3.1 Containment Joins

Every relational database system can evaluate the containment relation of the two node sets E, T by employing a standard join algorithm under the join condition: $pre(e) < pre(t) \leq pre(e) + size(e)$. However, even if both operands E, T are sorted on *pre-order*, the inequality condition does not allow to employ standard merge-join algorithms. Instead the join evaluation will be performed in nested-loop fashion, which requires $O(|E| \times |T|)$ comparisons, with $|E|, |T|$ denoting the cardinalities of the respective sets.

Special Containment Join Algorithms Several special containment join algorithms have been developed to overcome the problem. The *multi-predicate merge join* (MPMGJN), proposed by Zhang et al. (2001), advances cursors on the sorted input relations like a merge join. The cursors are used to tighten the inner loop with the aim to avoid unnecessary value comparisons. Grust and van Keulen (2003) showed further possibilities of pruning and skipping by exploiting tree properties of the pre-post relation. Their *Staircase join* algorithm ensures to read both input relations in single sequential scans. However, the Staircase join does not target the same operation as other containment joins. Instead of joining two arbitrary input node sets A, B resulting in a tuple set $\{(a, b)|a \text{ contains } b\}$, the Staircase join is designed to perform axis steps from a context set C resulting in all descendant nodes $\{d \in N|\exists_{c \in C}\ c \text{ contains } d\}$. Thus, the Staircase join in its initial form always uses the entire node set N as its second operand. Moreover, it performs duplicate elimination on the fly. If two nodes $c_1, c_2 \in C$ contain the same descendant d, it will get listed once in the result. Other containment joins output both tuples $(c_1, d), (c_2, d)$ instead. Whereas this behavior is desirable to execute axis step operations, it is not applicable in a scoring procedure, when elements and term occurrences need to be associated with respect to their containment relation.

Al-Khalifa et al. (2003) proposed two stack-based containment join algorithms that also ensure single sequential scans, but output all ancestor/descendant tuples either in sorted descendant or ancestor order. Algorithm 1 shows their so-called *Stack-Tree-Desc* join, though it is presented here differently. While again cursors move over both input operands as in merge joins, an additional stack keeps all ancestor candidates as long as further contained descendants can be found. In other words, every ancestor candidate is put on the stack when the cursor on the descendant list enters its region and

lives there until the cursor leaves its scope again. Hence, all nodes remaining on the stack are ancestors of the current descendant candidate. Notice, that all nodes on the stack stay at any time in ancestor/descendant relation to each other. Thus, the tree depth of the XML collection limits the size of the stack, which remains rather small for common XML data. The stack-based containment join has a time complexity of $O(|E| + |T| + |R|)$ for reading both operands and writing the resulting tuple-set R. While the outer loop iterates over the term occurrence set T, the total number of stack accesses lies in $O(|E| + |R|)$ for $|E|$ push/pop operations and $|R|$ read accesses.

Algorithm 1: Stack-based containment join algorithm

```
Stack-Tree-Desc(ancestor list A, descendant list D) ≡
  BEGIN
    initialize empty Stack and empty result list R
    t ← points (always) to node on Stack top
    a ← first node in A
    FOREACH d ∈ D DO
      WHILE t ≠ NIL and pre(d) > (pre(t) + size(t)) DO
        pop Stack
      WHILE a ≠ NIL and pre(a) < pre(d) DO
        IF pre(d) < (pre(a) + size(a)) THEN
          push a on Stack
        a ← next node in A
      FOREACH s on Stack DO
        append (s, d) to R
    RETURN R
  END
```

Retrieval-Aware Implementation Al-Khalifa et al. (2003) also showed on a set of simple and more complex path queries that their stack-based algorithm performs better then specialized merge joins, like e.g. the MPMGJN algorithm. When the containment join is employed, however, in a scoring procedure, the conditions change in some important aspects. Most often we want to score a large collection of documents or paragraphs by selective keywords. If the keywords would occur in too many documents they have stopword characteristics and do not contribute to the ranking. Therefore, we can expect in general that the cardinality of the element set outnumbers the term occurrences: $|E| \gg |T|$.

It is thus essential to skip unmatched elements as fast as possible. Notice, that the Stack-Tree-Desc join (Al-Khalifa et al., 2003) can be implemented in

two ways, traversing the ancestor and descendant candidate set in either outer or inner loop. The here presented version (Algorithm 1) exactly shows the wanted behavior by traversing the large element set E in the tight inner loop, which reduces the total number of value comparisons during the execution of the algorithm.

Simplifications for Unnested Ancestor Sets As mentioned already, at any time in the execution of Algorithm 1 all nodes on the stack stay in ancestor/descendant relation to each other. For unnested ancestor sets, the stack thus never holds more than one element and the algorithm could be simplified. Looking at typical structured queries, this case happens by far more often than containment joins with nested ancestor sets. The scoring of a node set follows in most cases a tag-name selection and nodes of the same tag-name are seldom nested in practice. Based on this argumentation, it seems worthwhile to especially recognize and support the case of unnested ancestor sets.

When the stack can be abandoned, the containment join falls back to a procedure resembling a merge join. Algorithm 2 shows the simplified procedure, further on called *Unnested-Tree-Desc* join. The inner loop forwards now the cursor on the ancestor list towards the element with the highest *pre-order* value $pre(e) < pre(d)$ without even fetching the corresponding *size* values.

Algorithm 2: Simplified containment join for unnested ancestor set

```
Unnested-Tree-Desc(ancestor list A, descendant list D) ≡
  BEGIN
    initialize empty result list R
    a ← first node in A
    a* ← next node in A
    FOREACH d ∈ D DO
      WHILE a* ≠ NIL and pre(a*) < pre(d) DO
        a ← a*
        a* ← next node in A
      IF pre(d) < (pre(a) + size(a)) THEN
        append (a, d) to R
    RETURN R
  END
```

Skipping by Binary Search Instead of moving the cursor from node to node as shown here, it is possible to forward the cursor by binary search, in order to skip large sequences of un-matching elements. Notice that this change

influences the time requirement of the algorithm. Whereas Algorithm 2 in the presented form has a data bound time complexity of $O(|E| + |T|)$, the binary search changes it to $O(log(|E|) \times |T|)$, which means an improvement in case $|E| > log(|E|) \times |T|$. In fact, the binary search can be implemented as a pure forward search with logarithmically decreasing steps over the element set E. In that case, our algorithm will never fall back behind the previous $O(|E| + |T|)$ even if the relation of $|E|$ and $|T|$ does not hold the above condition. Notice that $|R|$ is not mentioned anymore in the time complexity of the algorithm, since we know for unnested element sets that $|R| \leq |T|$.

Recognizing Unnested Sets The question remains how to recognize unnested element sets in advance. We cannot check all elements on a given node set whether they contain other nodes of the same set. However, by maintaining simple collection statistics and attaching *nestedness* as a property to the each node set, we can try to propagate the property by a few simple rules:

- Without further knowledge a node set is marked as *nested*.
- A node set passing a name test is marked *unnested*, if nodes of that tag name never occur nested in the collection.
- Child steps, parent steps, or filter predicates do not change the unnestedness of a node set.

Following these rules, we can sufficiently recognize a large extend of unnested sets, which allows to employ the above shown algorithm for a wide range of queries.

3.3.2 Experiments

In order to test the performance of the algorithms in a realistic scenario, the same corpus, query set and experimental setting was used as for the indexing tests (see Section 3.2.4) apart from a few changes mentioned in the following. Since the query length does not play a role here, only the short title-only queries were considered. The operand sizes, however, probably have a high impact on the containment join performance. So it is necessary to control their sizes. Instead of varying the tagname of the requested elements among the 50 queries, the element set was kept fixed here. In a second experiment special single term queries were executed, to also control the size of the term occurrence set. Index schema I (see Section 3.2.3) was employed in all tests since the Staircase join would not work on the other and we wanted to show its timings as a reference to non-retrieval-aware containment join implementations.

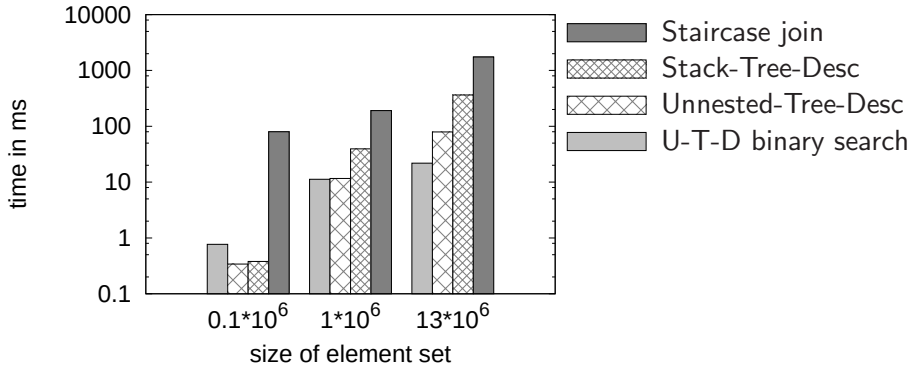


Figure 3.6: Performance depending on the size of the element set

The Staircase join was used here in a special loop-lifted variant (Boncz et al., 2006), which allows to return ancestor/descendant tuples by using the iteration column for listing all ancestor candidates.

Size of the Element Set In a first experiment, the influence of the element set size was tested. The system had to score 3 different element sets on all 50 queries: (1) a randomly sampled subset documents tagged `DOC` – 0.1×10^6 nodes, (2) the set of all documents – 1×10^6 nodes, (3) the set of all paragraphs tagged `P` – 13×10^6 nodes. All three sets represent typical retrieval tasks. The smallest set (1) demonstrates the case, where a node set is coming from a preselection, e.g. the scoring of all documents having a certain date. The other two sets represent a typical document (2), respectively paragraph ranking (3) task.

The applied query plan P1 (see Section 3.3.3) executes one containment join for the occurrences of each single query term. Therefore, 132 containment joins are performed in total for executing all 50 multi-term queries. The average execution time of one containment join is shown in the log-scale histogram (see Figure 3.6). Obviously, the more *retrieval-aware* implementation of our Stack-Tree-Desc join shows already a high performance improvement over the Staircase join. In all 3 cases it needs at most 50% of the execution time compared to the Staircase join. Since all tested element sets are unnested, also the special Unnested-Tree-Desc join can be employed. Here we observe again a performance win by an order of magnitude for large element sets. In case of the smallest element set, the timings have to be taken with care. Firstly, the timings fall below the measurable units, and secondly, we probably observe here caching effects of previous operations. Still, the outcome for all three element sets confirms nicely the theoretically ex-

pected linear dependence between element set size and execution time. Only the Unnested-Tree-Desc join with binary search shows a different behavior. Whereas it suffers from the search overhead on the smallest set, its execution time grows only sub-linear with the size of the element set.

Size of the Term Occurrence Set The second experiment kept the element set fixed while varying the size of the term occurrence set in a controlled way. The set of all documents (2) was used as the element set. In order to choose realistic query terms from varying selectivity, all tested terms were taken from the query set starting with the one having the highest number of term occurrences and ending with the most selective term. In increasing order of selectivity, the following 4 terms got selected: (1) *American* – 370×10^3 occurrences, (2) *storms* – 37×10^3 occurrences, (3) *extinction* – 3.7×10^3 occurrences, (4) *WTC* – 12 occurrences.

Figure 3.7 shows the corresponding execution times. Obviously, both stack-based joins are less influenced by the size of the term occurrence set. The way larger element set clearly dominates here the execution time. The Unnested-Tree-Desc join, on the other hand, shows a clear benefit of smaller term occurrence sets. Already the version without binary search brings its execution time for the smallest set down to 20% of the time required for the largest term occurrence set. The difference to the Stack-Tree-Desc join can be explained by the less look-ups for element sizes. Employing binary search, the size of the term occurrence set even becomes the determinant factor of the execution time. The experiments confirm also the theoretically deduced turning point where binary search becomes less efficient. For the largest term set, the condition $|E| > log(|E|) \times |T|$ does not hold anymore and in fact we observe the algorithm without binary search to show a slightly better performance in this case.

3.3.3 Query Plans

Database systems process queries on different layers (see e.g. Elmasri et al., 1999, Chapter18). A query is transformed into an algebraic expression on the logical layer and later mapped to the best available operator implementation on the physical layer. Typically, query optimization takes place on all these layers considering alternative but semantically equivalent query plans, that define the order of operations. Query optimization uses either simple heuristics or applies cost models to estimate which plan executes the query most efficiently. We will take in the following a look at the translation of the scoring procedure into an efficient sequence of database operations. From

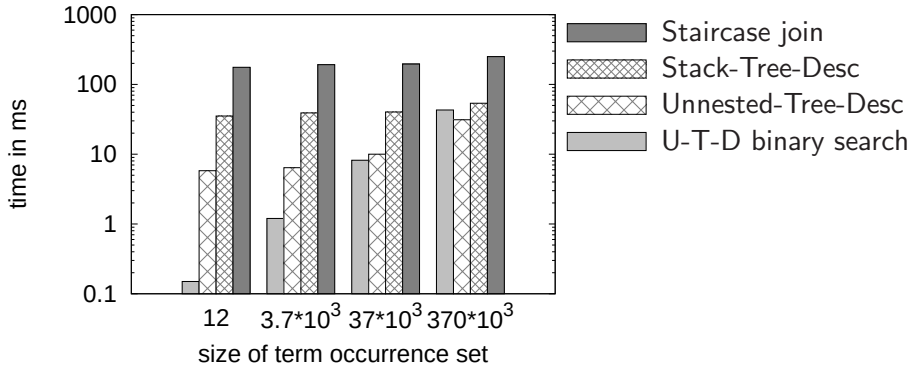


Figure 3.7: Performance depending size on size of the term occurrence set

the point of view of our XML retrieval system, which uses the DBMS only as an execution back-end, the considerations take place on the physical level.

Although the containment join evaluation plays a major role in the procedure of scoring of elements, other operations – as listed in the beginning of this section (3.3) – have to be considered as well when we try to optimize the system's performance. In contrast to the containment join, the term selection, score calculation and aggregation are, however, well supported by standard database operations and the index. Thus, we are left with the remaining task to investigate efficient query plans.

Figure 3.8 shows the alternative plans that will be discussed in this section. Before we actually compare the different plans, some notations need to be introduced. All plans start with node selections for terms σ_{T_i} and elements σ_E. In fact, we only show plans for a scoring procedure with two query terms selecting the sets T_1, T_2. It is, however, easy to imagine how the plans would look like for n terms. The selections are followed in all three cases by a containment join $\bowtie_{(contm)}$. The join outputs all (e, t) tuples, but for the later score calculation the number of occurrences of all different unique tuples has to be aggregated $\{\texttt{cnt}\}_{(e,t)}$. In case that the sub-branch of the query plan ensures that only tuples of a certain term t are expected, the aggregation algorithm can be simplified to $\{\texttt{cnt}\}_e$. The same holds for the following arithmetic operations `[*/]` to calculate the score portions for each unique tuple (e, t). The actual arithmetic operations are determined by the employed retrieval model. Different retrieval models do not only involve varying arithmetic operations, but also require different element and term statistics. The execution times will therefore vary depending on the employed model. The final score aggregation per element is denoted here by $\{\texttt{+}\}_e$, since for many retrieval model log-based partitial scores have to be

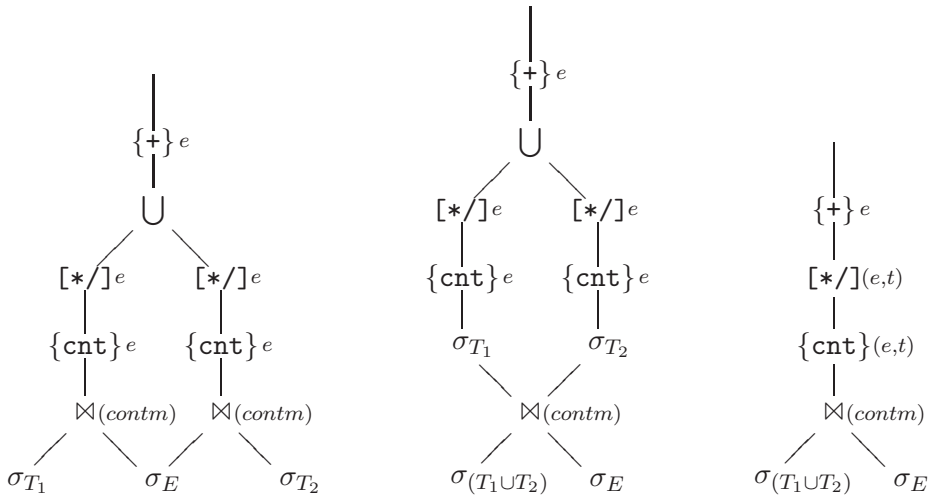


Figure 3.8: Alternative Query Plans

summed up.

It should also be remarked in this context, that our database back-end *MonetDB* does not make use of so-called pipelining techniques, but follows a bulk processing approach (Boncz, 2002). Hence, each operation is executed completely materializing its intermediary result in memory before the next operation starts processing the data.

As we see in the 3 suggested plans, the order of operations stays constant in all three cases. Although changes would be possible here, it is easy to argue why improvements cannot be expected by order changes. It is always preferable to start with the selections on terms and elements, instead of performing the containment join on the entire node set. The subsequent tuple count and arithmetic operations require the result of the containment join. It is not strictly necessary to perform the aggregation {cnt} before starting the retrieval model dependent score calculation [*/], but it is advisable to reduce the intermediary result set as soon as possible. The score aggregation, finally, cannot be pushed down since it again requires partial scores to be calculated.

Looking at the differences, the first plan P1 executes $\bowtie_{(contm)}$, {cnt}, and [*/] in an independent branch for each query term, while the third plan P3 employs single operations for all steps. Obviously, the number of involved operations for P1 is considerable higher, especially for a growing number of query terms. On the contrary, the operations involved in P3 have to work on unique (e, t) tuples rather than performing {cnt} and [*/] on unique elements only. This little difference makes the employed algorithms more complex and expensive. Moreover, working with an DBMS with full vertical

fragmentation like *MonetDB* becomes a disadvantage here. Aggregation on unique tuples involves assigning them to single identifiers and several mapping joins. The same problem arises for the calculation of partial scores thereafter. In consequence, the advantage of the lower number of operations might disappear by the increased complexity of the remaining ones. Plan P2 should be seen as an intermediary solution. The containment join is executed once only, but the branches are still split thereafter to enable the less complex score calculation per query term.

Theoretically, we can also imagine a query plan with separate operator branches for each element that has to be scored, similar to the splitting on query terms. The complexity reduction would be the same and the final aggregation could be simplified as well, however, due to generally large number of elements, the total number of operations becomes unacceptably high.

3.3.4 Experiments

The above discussion showed already some advantages and disadvantages of the proposed 3 query plans, however, it requires empirical testing to achieve a clear comparison of their performance. As before, the Aquaint corpus and the 2005 Robust Track query set were used for testing. The further test setup has been described in detail in Section 3.2.4. Instead of reporting only total execution times for the different query plans, the measurements were split over the involved operations. For those operations the times were summed up during the 50 queries and later reported as averages for the evaluation of a single query. If a certain operation is performed in several branches of the query plan, the shown execution time is related to the total time for executing the operation in all branches.

Performance on Short Queries Figure 3.9 compares the query plans when evaluating the 50 short title-only queries. The plans P1a and P1b are both instances of the above presented query plan P1, but differ in the employed aggregation method as described later. It will become clear then why both versions are shown here.

We can first of all see that plan P1a and P2 are almost equally efficient and clearly outperform the other two. The stacked histogram also shows how the execution time is divided over the involved operations. In fact, the distinction is not in all cases as clear as one might expect. The operations often involve sub-operations that could be assigned to several parts. For instance, the containment join requires both operand sets to be pre-order sorted. The term selection of P1a/b directly returns the terms pre-order sorted from the index. However, when selecting at once the term occurrences of all query

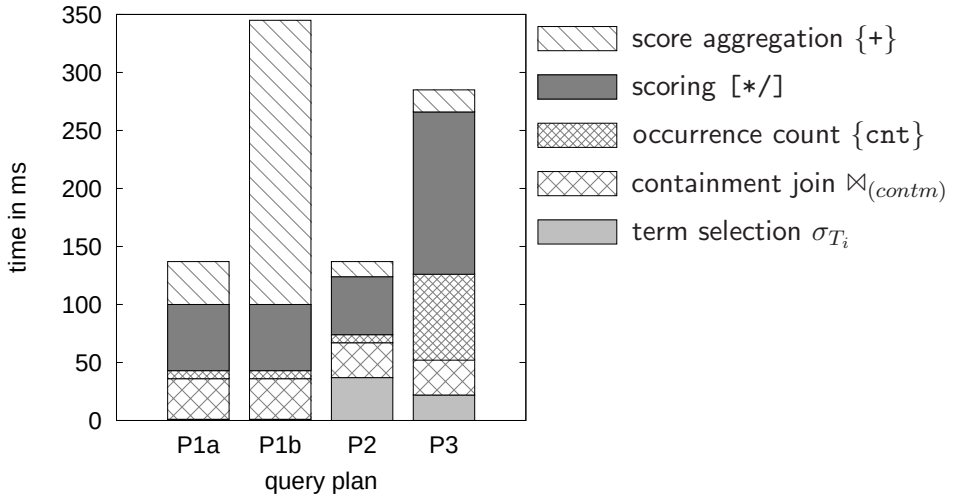


Figure 3.9: Performance depending on the used query plan for short queries

terms as in P2/3, a sort operation is required on the unified set. The time of the additional sort operation could be assigned to the term selection part with the argument that all selections should satisfy the same output properties, but we could also assign the sorting time to the containment join that requires the sortedness of its input. We decided here for the first option, which can be seen in the considerably higher selection times for plan P2/3.

Looking at the containment join part, we can first observe that the employed tree-merge join for unnested element sets (Algorithm 4 of section 3.3.1) is not causing a high overhead in the entire scoring procedure. If a Staircase join is employed here instead, the containment join part would clearly dominate the overall execution time. Furthermore, the comparison of the plans P1a/b and P2/3 shows that for short queries the multiple execution of the join for each single query term is only little slower than the joined execution.

The first aggregation for counting the occurrences of all unique (e, t) tuples is neglectable in all plans apart from P3. The complexity overhead of the joined computation clearly becomes a disadvantage. The same observation holds for the score computation part of the plans. As an example of the more complex calculation, we can think of the simple multiplication with terms collection likelihood. While this value is a constant factor for all tuples in single branch of P1/2, the multiplication in P3 has to perform an implicit join on the terms t to use the right factor for each (e, t) tuple.

Finally, the score aggregation part shows different timings in all 4 plans though the differences are not apparent in the plans themselves. The reason lies again in the different physical operators employed by the plans. In P3

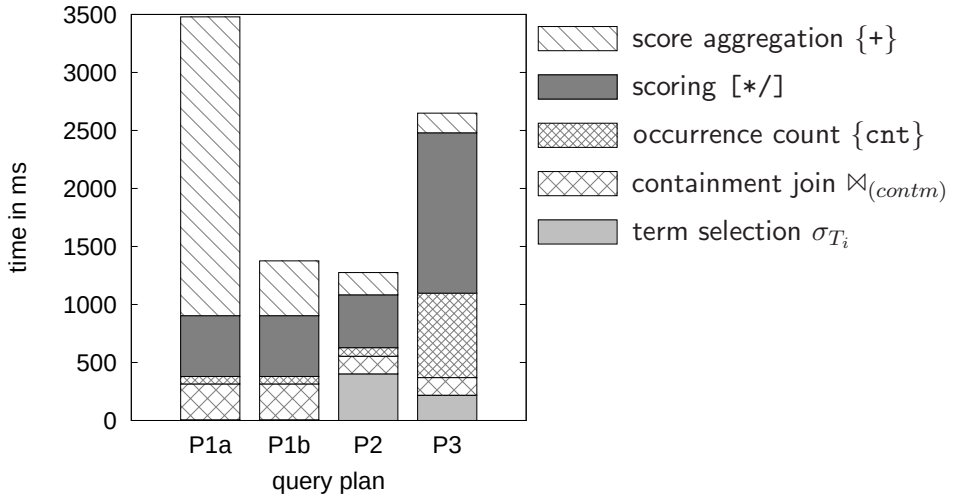


Figure 3.10: Performance depending on the used query plan for long queries

we do not need to gather partial results and can directly employ a standard database aggregation operator. For all other plans, scores are directly accumulated when joining the results of two branches. We can either create an accumulator BAT for all elements beforehand where the partial score values are added to (P1b and P2) or we execute a union and aggregation for each branch (P1a). In the later case, each union operator requires to copy all values to a new result BAT. In the first case, however, the accumulator BAT have to keep a score slot for all elements, also those that have not been assigned a partial score, yet. Here, plan P2 has an important advantage over P1b, since it is known after the single containment join which elements contain at least one of the keywords and thus really require a score slot in the accumulator BAT. In plan P1b, this information is not available and the accumulator BAT has to stay considerably larger. The performance impact of these differences is clearly visible in Figure 3.9. At least for short queries, it plainly suggests not to use the accumulator BAT in plan P1.

Performance on Long Queries Although typical user queries rarely contain a large number of keywords, experimental setups more often require to rank element sets by longer queries. It is interesting here to see how the different query plans react in the later case. Obviously, the branching factor of P1 and P2 gets considerably higher here. The above experiment was repeated but instead of using the topic title, its longer narrative field was used to construct the 50 queries. The results are shown in Figure 3.10.

Looking at the total times, plan P2 is slightly ahead all others now, and

plan P1b and P1a have changed places. When we might have reckoned that a higher branching factor have a highly negative performance impact, this expectation is met only partly. Plan P3 is still considerably slower then the best performing ones. The ratio of execution times between the operators inside the plans roughly stayed the same as before. Only the containment join fraction is diminished in P2/3. The other operators apparently are not negatively influenced by the branching. These two observations explain the slight advantage of plan P2. It combines the single containment join execution with the faster branched score computation. Both favorable concepts brought together even compensates here for the additional term selection time when splitting the results of the containment join again.

The figure also shows why plan P1 is shown with both different score aggregation methods. Although the plan has the potential to compete with best performing P2, the union and aggregation becomes the major issue here. Both tested physical operators fail to give a fast response time for either short queries as shown in Figure 3.9, or for long queries as shown in Figure 3.10. Hence, for working with P1 it is necessary to either develop a new aggregation method not suffering from the seen shortcomings, or to use dynamic optimization techniques to choose in each case the best physical operator for score aggregation.

3.4 Efficiency/Effectiveness Tradeoffs for Complex Queries

In this last section we want to move from the evaluation of simple queries to more complex ones. Looking back at one of the initial examples for structured querying, we can find the following NEXI query:

```
//html[about(., ir db)]//p[about(., xquery)]
```

The query consists of two simple sub-queries, `e1[q1]` and `e2[q2]`, which can be evaluated using the techniques introduced in the last sections. However, the example query here requires more than the subquery evaluation. It asks for paragraph elements `p` whose relevance is determined by the combined evidence of containing the term "*XQuery*" and being included in `html` elements about "*DB*" and "*IR*". Hence the scores of the first subquery need to propagated to the level of the contained `p` elements and finally the scores of both subqueries need to be combined.

All involved operations can be formalized in an algebra of scored elements, called *score region algebra* (Mihajlovic et al., 2005). We will not formally

introduce the algebra here, but remark that the steps in the above described evaluation schema directly correspond to operators of this algebra, namely

(1) a *selection* operator for elements given a tag-name,
(2) a *scoring* operator to score elements by a term query,
(3) and a *propagation* operator to propagate scores down to included elements.

All those operators work on scored element sets. They always return scored element sets and apart from the selection operator they also take scored element sets as their input operands. If an operator assigns new scores to a scored element set, the old scores are combined with the new ones. Mihajlovic et al. (2005) show that their algebra consisting of a few more operators than the three here shown is able to express arbitrary complex NEXI queries.

Strict vs. Vague Query Semantics Thinking in terms of XPath, the base of the NEXI language, the scoring clause is always embedded in a predicate `[about(., t1 ... tn)]`. A predicate is evaluated to a boolean value for each element in a given node sequence, causing those elements to pass the filter that satisfy the predicate. Consequently, the scoring clause has to be evaluated to a boolean value as well. If we want to abuse the predicate for the pure side effect of assigning scores to the elements, the effective boolean value of the scoring function should be set to `true` in all cases. This is clearly not the only possible query interpretation. We can require that an element has to reach a certain score threshold in order to satisfy the predicate condition. The least strict setting of such a threshold would be to filter out all zero scored elements. In other words, the `about` function would assign a `true` value to all elements that contain at least one of the query terms. The example query helps to demonstrate the consequences of the different query semantics. Whereas the *vague* interpretation allows to return paragraphs that are highly relevant to "*XQuery*" even when the surrounding html documents does not contain the terms "*DB*" or "*IR*", the same paragraph would not qualify according to the *strict* interpretation. With respect to the retrieval quality, we can expect the *vague* interpretation to work more recall oriented. It delivers a larger result set and thus increases the chance to find relevant documents among those not returned using *strict* semantics. If the score propagation and combination is modelled appropriately, the retrieval precision of both semantics should roughly stay the same, since we expect the top scoring elements to satisfy both constraints. The advantage of the *strict* interpretation lies mainly in query performance. Whereas simple queries are not affected, complex queries highly profit from the filtering effect, that

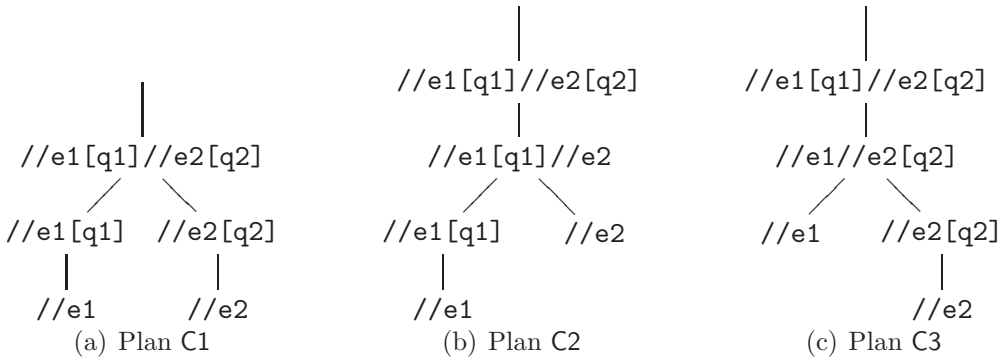


Figure 3.11: Different Plans for a Complex Query

significantly reduces intermediary result sizes.

Query Plans for Complex Queries Once having a query expressed in algebraic form, a query optimizer can consider different execution plans. Figure 3.11 shows 3 different options to evaluate a query of the form shown in the example. Compared to the query plans considered in the previous section, the optimization in this case is situated on the logical layer. We used here NEXI syntax describing the outcome of a certain operator to avoid the introduction of an algebraic notation.

Plan C1 executes the two simple queries isolated from each other and later propagates and combines the scores at the requiered output level. Plan C2 and C3 first evaluate one of the simple queries, propagate the results to the second element set, and finally rescore this set according to the term query and combine both scores. Plan C2 and C3 seem completely symmetric in the presented form, however, the execution of C3 is in fact more sophisticated. Notice that the query defines a final output target, here the set of paragraph elements. In order to express plan C3 as close as possible in score region algebra we would need to propagate the scores first up to the level of html documents and later down again to satisfy the output requirements.

Considering alternative execution plans is beneficial only, if one plan is expected to outperform the other. In general we will prefer plans that produce smaller intermediary results, especially as the input of costly operators. In this case, the up/down propagation of scores is almost as costly as the scoring operator. The propagation also involves a containment join and score aggregation. On this background, we cannot apply heuristics to push down the cheaper operator in the query plan. However, we show that for the given query pattern even a simplistic cost modeling is sufficient to find the better plan in most of the cases.

Simple Cost Modeling We want to demonstrate here shortly the effect of query optimization for complex queries. The cost modeling introduced in the following is neither meant as a generic approach nor trying to capture all cases and effects. However, it shows on the analyzed example query pattern how to find with simple means the best query plan.

The analysis of the scoring operator has shown that the performance was mainly influenced by the two operand sizes, namely the cardinality of the element set and the term occurrence set (see Section 3.3.3). Similarly, the performance of score propagation operator is determined to a large extend by the sizes of both input element sets. We will thus model propagation and scoring times T by:

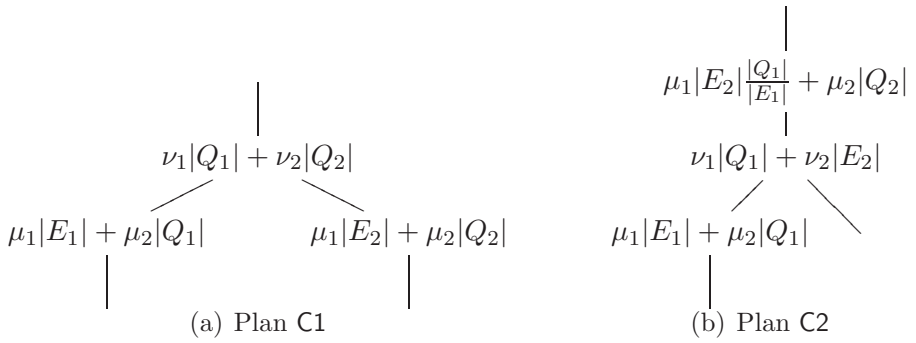
$$\begin{aligned} T(E\texttt{[}Q\texttt{]}) &= \mu_1|E| + \mu_2|Q|, \\ T(E_1\texttt{//}E_2) &= \nu_1|E_1| + \nu_2|E_2|. \end{aligned}$$

The set Q denotes the union of all query term occurrences. E_1, E_2 represent the operand sets of the score propagation, both being element sets with the respective tag-names `e1` and `e2`. Notice, that the operand sets do not necessarily contain all nodes of the given tag-names, especially not when the operator occurs further up in the query plan. The parameters μ, ν model the influence of the operand sizes on the execution time of the respective operator.

Cost modeling for complex queries also requires the estimation of intermediary result sizes. In case of the scoring operator it depends first of all on the applied query semantics. When using a *vague* query interpretation, the result size is equal to the size of the element set that is getting scored. In case of the strict interpretation, we know that the result size can at most be as large as the number of term occurrences if no element contains more than one term. Assuming that this number is clearly lower than the element set size, we use it as the best possible approximation: $|E\texttt{[}Q\texttt{]}| = |Q|$.

In case of the score propagation $E_1\texttt{//}E_2$, the result size would be equal to the number of elements E_2, if all elements from E_2 are contained by elements from E_1. Assuming that the user roughly knows the schema of documents in the collection, we expect all nodes with tag-name `e2` to occur inside nodes with tag-name `e1`. Based on this simplifying assumption, the result size of the down propagation operator is smaller than $|E_2|$, only if the first operand E_1 does not contain all nodes of its tag-name, thus if E_1 is only a fraction of the complete tag-name set `//e1`:

$$|(\texttt{e1//e2})| = |E_2|\frac{|E_1|}{|\texttt{//e1}|}.$$



(a) Plan C1 (b) Plan C2

Figure 3.12: Cost Modeling of C1 and C2 for strict query semantics

The simple cost modeling shows immediately that the plans C1 and C2 are equally efficient, when using *vague* semantics. Our model assigns the same operand sizes to all the corresponding operators in this case. The picture changes, when using *strict* query semantics. Figure 3.12 shows the application of our cost models to the plans C1 and C2. The element sets E_1, E_2 equal the full tag-name sets `//e1`, respectively `//e2`. In Plan C2 the result size of the first scoring operator is estimated by $|Q_1|$. The following down propagation thus returns only a fraction of the full set $|E_2|$, approximated by $\frac{|Q_1|}{|E_1|}$.

Summing up the costs of all operations, our estimation shows that C1 performs faster than C2, if

$$\begin{aligned} \mu_1|E_2| + \nu_2|Q_2| &< \mu_1\frac{|Q_1|}{|E_1|}|E_2| + \nu_2|E_2| \\ \Leftrightarrow \frac{|Q_2|}{|E_2|} &< \frac{\mu_1}{\nu_2}\frac{|Q_1|}{|E_1|} - \frac{\mu_1}{\nu_2} + 1. \end{aligned}$$

Since we expect that the parameters $\mu 1$ and $\nu 2$ not to differ highly in practice, the estimation might even be simplyfied to $\frac{|Q_2|}{|E_2|} < \frac{|Q_1|}{|E_1|}$, which provides an easy to calculate preference indicator.

Plan C3 is not considered in this context, since it cannot be translated to the introduced operators in a straighforward way. If an additional up and down propagation of the scores has to take place, the plan will not reach the efficiency of the other two.

3.4.1 Experiments

In contrast to the other experiments in this chapter, it is necessary to test the performance of complex queries. Moreover, we need to evaluate the retrieval

quality going along with the usage of strict or vague query semantics. The INEX Wikipedia collection and Adhoc task provides both, a large enough number of complex user queries and associated relevance assessments to measure the retrieval quality.

The Wikipedia collection was already used in this chapter for indexing experiments (see Section 3.2.4). The collection consists of encyclopedia entries, whose original web wiki mark-up is mapped to XML tagging, resulting in large corpus of neatly structured articles. The Adhoc task 2006 comes with a total number of 125 query topics, all of them expressed both in natural language and by a NEXI query. Since we were interested in complex queries only, we first selected those queries including at least two scoring predicates with an `about` function on different element sets. The task definition regards the NEXI query as an approximation of the intended user interest. It therefore allows the assessors process to mark elements as relevant to the topic that cannot be selected by the corresponding NEXI query. We found a high number of elements marked as relevant that do not match the tag-name of the required target element in the corresponding NEXI query. Although it is interesting to study the users ability to express an intended information need appropriately in the NEXI language, such unreachable but relevant elements would only reduce the recall measures of tested XML retrieval system. We therefore filtered out all assessments of elements that do not match the query target. Consequently, a few further queries were disselected, since they had no relevant answer any more. After that filtering process, we were still left with a number of 50 complex queries, which were used to compare the retrieval quality in our experiments.

For the performance measurements, we selected 25 queries that directly match the analyzed query pattern: `e1[q1]//e2[q2]`. We further tried to achieve a variance on selected tag-names among the queries, to avoid caching effects and to test the cost models in different situations. Since most of the queries select `article` and `section` elements, we preferably picked queries asking for others tag-names and mixed them in a way that consequtive queries do not select both times the same element sets.

Retrieval Quality We used PF/Tijah to find the 1500 most relevant answers to each of the 50 selected queries. The system was set to employ the *NLLR* retrieval model. Unlike the INEX community, we used mean average precision (MAP) and precision at 5 or 10 retrieved elements (P@5/10) as the main quality measurements, since they were also used at other places in this thesis and represent established retrieval quality indicators. An external comparison of our results with other INEX participants is anyway not in the

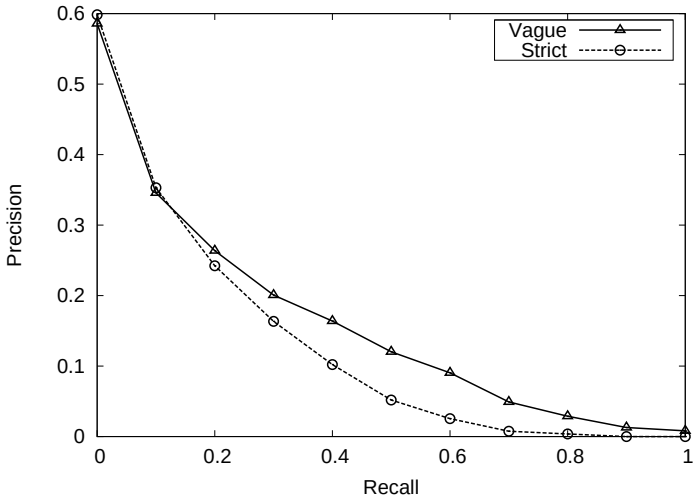


Figure 3.13: Precision/recall graph when applying different query semantics

interest of this particular evaluation. Looking at the retrieval quality, it is not necessary to compare the different query plans, since they deliver the exact same results.

	Vague	*Strict*
MAP	0.146	0.115
P@5	0.364	0.392
P@10	0.320	0.336

Table 3.3: Main Quality Measures

Table 3.3 shows the influence of the chosen query semantics on the retrieval performance. The MAP values demonstrate the expected advantage of the vague interpretation with respect to recall. The retrieval precision, on the other hand, does not suffer at all from the early filtering. In fact, the strict interpretation even beats the vague one on both early precision values. The precision advantage, however, remains small which is better visualized in a complete precision/recall graph as presented in Figure 3.13.

Performance PF/Tijah does not allow to influence the query plan generation. Therefore the plans for the 25 test queries had to be generated manually to test the performance differences between plan C1 and C2.

Table 3.4 gives a first overview on the average query performance for both tested plans and query interpreations. Evidently, the application of *strict* query semantics leads to a better query performance. The average total execution is more than 4 times faster than in the case of a *vague* interpretation. The different query plans, on the other hand, do not show a clear advantage of the one or the other when looking at their average timings. On the level of involved operations, the expected differences are clearly

	Vague		*Strict*	
	C1	C2	C1	C2
`e1[q1] + e2[q2]`	178.4 ms	228.7 ms	90.3 ms	67.8 ms
`e1//e2`	250.0 ms	198.0 ms	6.5 ms	33.8 ms
total	429.0 ms	427.3 ms	96.9 ms	102.0 ms

Table 3.4: Performance of C1 and C2

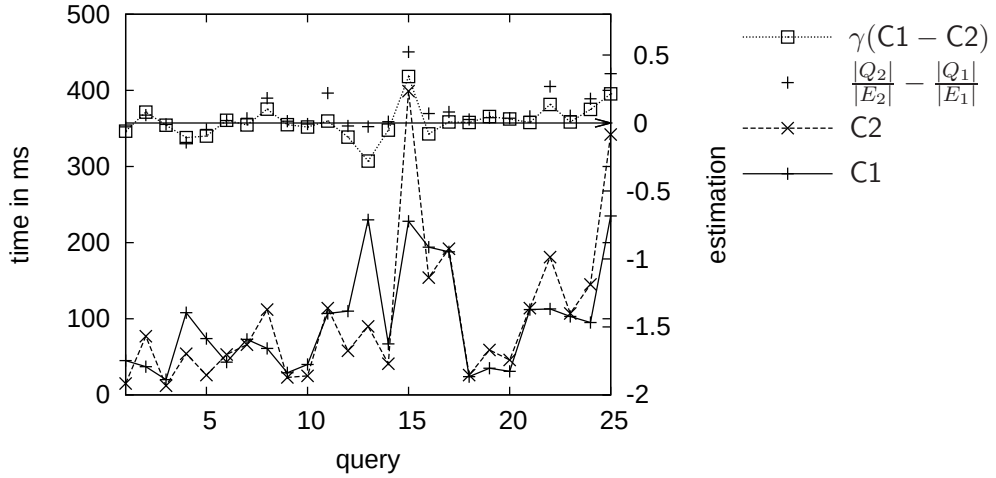


Figure 3.14: Real Timings and Estimations per query

visable. The score propagation in C1 either suffers from the large element sets in the *vague* case or profits from the largely filtered input in the *strict* case compared to C2. The total timing, however, stays for both plans the same.

The picture changes if we take a look at the individual query timings using *strict* semantics (see Figure 3.14). Here we see not only a high variation on the execution time per query, but also clear plan specific differences for a number of queries. Plan C1 and C2 alternately beat each other. Hence, none of the plans should be preferred in general. The figure also displays the outcome of our cost modeling. Since we are interested in a binary decision for either C1 or C2, the difference $\frac{|Q_2|}{|E_2|} - \frac{|Q_1|}{|E_1|}$ is calculated. A positive outcome means that C1 should be preferred, and vice versa for a negative difference. Although it is hard to see in the graph, our simple query optimizer fails only in 4 cases, where the differences between both plans remain small anyway. In order to demonstrate the estimation quality, we overlayed the estimation by the down-scaled real time difference between the execution of C1 and C2. The factor γ is only used to display both differences in the same scale. The

overlay shows that our cost modeling is quite accurate. When the real time differences are high, we also see a higher indication for the respective better plan in the estimation.

	strict
C1	96.9 ms
C2	102.0 ms
optimized	82.2 ms

Table 3.5: Optimized vs. non-optimized execution

Finally, it is interesting to calculate the profit of query optimization. If we neglect the short runtime of an optimizer itself, the time gain can be simulated by adding up the timings of those plans indicated as less costly. The calculated average execution of an optimized system time significantly beats a system using always one of the suggested plans only (see Table 3.5).

3.5 Summary and Conclusions

We opened this chapter by taking a look at the use of structural features in text retrieval and extracted a list of minimal requirements for query languages that allow to query content and structure. On this background, we compared the two most popular query languages for structured retrieval on XML, namely NEXI and XQuery FT. Both followed different design goals and consequently show a number of differences mainly with respect to expressiveness and simplicity of use. The advantages and disadvantages of both languages have been discussed leading to the proposal of an intermediary approach that suggests a NEXI embedding in XQuery, extending the database-style query language by full-text search functionality. The proposed language embedding combines a number of advantages from both base languages. It extends the expressiveness considerably by enabling the free composition with XQuery statements, while still keeping the text search functionality as simple and semantically "safe" as in the pure NEXI language. Unfortunately, the proposed language embedding leads to overlapping expressiveness and disables static query compilation.

In the remainder of the chapter, we focussed on performance issues investigating three aspects of the execution of structured queries, namely index support, efficient algorithms, and query optimization. After identifying the basic operations involved in the execution of structured queries, we first addressed the design of an index structure that maintains all necessary information and supports the efficient execution of the basic operators. Inverted indices known from document retrieval are not appropriate for structured retrieval due to high redundancy when listing term occurrences of nested elements. Our index structure overcomes the redundancy problem by making use of a *pre/size* encoding that captures the XML structure. In particular,

we showed how such an encoding can be maintained in a database system and proposed two variants for the physical storage of the size information. Both variants have been tested with respect to space and performance efficiency, demonstrating the clear advantage of the second proposal for the most common type of keyword queries on a tag-name specified element set. For further improvements in the indexing domain, it would be necessary to employ and test light-weight compression techniques to further reduce the amount of stored data and enable faster access.

The execution of the often occurring base operations also asks for efficient algorithms. Apart from the containment join, most other operations are supported efficiently by the database system. Hence, we examined different recently proposed containment join algorithms and introduced adaptations for a more retrieval-aware implementation with the intention to tighten the loop traversing the large element set. Especially, we simplified the algorithm for the common case that the set of elements is not nested itself. The simplified version also allowed to employ binary search when traversing the often large element set. All considered versions of the containment join operation have been tested with respect to their performance, showing that the simplified algorithms achieve high performance improvements, whenever they are applicable. The version employing binary search has to be used carefully. Though it reduces the execution time considerably on small term occurrence sets, the binary search causes overhead when the size of the term occurrence comes close to size of the element set.

We also examined query plan optimization on two different levels. First, we focussed on query plans for the execution of term queries on a given element set, consisting of basic database operations. Thereafter, we also examined more complex queries that combine the results of several simple term queries on different element sets. Query optimization on this higher level was only touched on by analyzing optimization on a specific complex query pattern. A more complete study is still required but out of the scope of this work. Testing the different plans on low-level optimization showed first of all the cost ratio of all involved base operations. Especially score aggregation can easily predominate the total execution time, when the wrong aggregation method is chosen. The tests further showed that the number of query terms had a clear impact on the choice of the best query plan. Although no single query plan was better than the others in all given situations, one of them showed a more robust high performance. Another one can be excluded, since it never came close to the best execution times.

In case of the higher level query optimization, both considered plans showed a similar average performance but differed in their execution costs when looking at the individual queries. We showed on the analyzed example

query pattern that a simple cost model allows to find the better plan in most cases. Hence, a dynamic optimization strategy could considerably cut down the execution costs. In order to use query optimization on a wider range of complex queries, we would need to extend the cost modeling framework.

Finally, we analyzed a strict and vague query interpretation with respect to their retrieval quality. The two different query semantics also showed a considerable impact on the execution time. We could demonstrate that the strict interpretation does not only improve the performance due to early filtering, but also keeps an equally good or even higher precision on the top of the result list. The vague interpretation only showed a better recall. Hence, if the application of the search system allows to sacrifice a high recall, the user profits from the strict interpretation due to its faster execution while the first result page still shows items of the same quality.

The fact, that finding the most appropriate query semantics is still an issue of research, shows also how far the current query languages for structured retrieval are away from being used as an intuitive query language for the end-user. If we aim at retrieval systems for non-expert users that allow to incorporate structural features in the search specification, more end-user oriented languages or user interfaces have to be developed. Moreover, the retrieval systems have to learn more about the semantics of structure. An XML retrieval system does not make a principal difference between the markup of a title or footnote, though the later provides in general less relevance evidence.

4

Entity Retrieval

When people use retrieval systems, they are often not searching for documents or text passages in the first place, but for some information contained inside. Many information needs can be described by one of the two following patterns:

(1) The information seeker knows of the existence of a certain person, organization, place – in general *entity* – and wants to gather any kind of information about it. For instance, someone is searching for more information about a special sickness.

(2) The searcher wants to find existing persons, organizations, places – *entities* – of a certain type, e.g. looking for hairdressers in his/her home town.

Combinations of both types are common as well. We can think here again of a person looking for hairdressers, but also for their location and for a rough impression of their typical customers.

The two general patterns are not given here with the aim to define a new taxonomy of information needs like the well-known categorization of web search tasks by Broder (2002), but to motivate the need of entity retrieval. In fact, the term "entity" is used here in the description of both patterns. However, in the first case the entity is known already and only information about it is searched, whereas in the second case the entities are unknown and have to be retrieved. While queries of the first type are appropriately addressed by keyword queries in standard retrieval systems, queries of the second type should be handled in a different way, as we will show in this chapter.

The chapter is structured in the following way. We first define and describe entities and the respective retrieval tasks, then take a look at existing approaches in general terms. Entity Containment Graphs will be introduced

as a suitable way to model relationships between entities and text units. The graphs are used later to propagate relevance between its vertices. This part is based on earlier published work (Rode et al., 2007; Tsikrika et al., 2008; Rode et al., 2008). Finally, we investigate special issues around ranking entities of different types, presented also in our recent article (Zaragoza et al., 2007b).

4.1 Entity Retrieval Tasks

When speaking about *entity* retrieval, which is a relatively new field of retrieval research, we first need to define the task more clearly. According to the corresponding Wikipedia entry, "an entity is something that has a distinct, separate existence, though it need not be a material existence."[1] We further require here that those existences are given a *name*. Hence, we are interested here in so-called *named entities*. Named entities can be categorized by their *type*. Some examples of such types have been given in the introduction, like persons, organizations, and locations. The set of types is theoretically unlimited, but in most practical cases bound to a predefined set that can be automatically recognized by the available tools (e.g. named entity taggers) or limited to those types that are of interest to a certain application. When working with rich fine-grained type sets, we will often use a hierarchical organization of types. In that case, entities have one or more types and subtypes, e.g. an "apple tree" being of type `plant`, `tree`, and `garden plant`.

Each entity has its own identity. Unfortunately, neither names are unique nor the combination of both name and type. Everyone knows different entities of type `person` having the same name. The problem gets worse when we think of the work of automatic taggers that have to recognize named entities by various *mentions*, like abbreviations, nicknames, spelling variations in different languages, or by pronoun references (coreference resolution). An overview on such information extraction tasks and the respective problems is given by Cunningham (2005). Amongst others, Chen and Martin (2007) present a recent approach towards name disambiguation of entities. The paper also shows the syntactical and semantic features used for this task. In order to assign appropriate fine-grained types to the entities, Kazama and Torisawa (2007) demonstrate how external knowledge like the Wikipedia corpus can be employed to improve the type labeling. Though named entity recognition is an important requirement for entity retrieval as we discuss it

[1] see `http://en.wikipedia.org/wiki/Entity` as of 2.12.2007

in the following, it will not be studied in this work. We simply assume the availability of automatic or manual entity tagging on the text corpus, and leave the research in this field to the information extraction (IE) community.

Having defined what we mean by entities, entity retrieval is regarded here as the task to retrieve and rank entities according to their relevance to given query. As we show later, we can distinguish several subtasks in the domain of entity retrieval, but all can be addressed by the same general approach.

Entity vs. Structured Retrieval Coming from structured retrieval it is important to point out the differences and similarities. The following NEXI query also returns a ranked list of entities assuming that it is rooted on an entity tagged text corpus:

```
//DOC[about(., Pablo Picasso)]//entity[.//@type = "location"].
```

However, the query will return the same entity multiple times in case it is mentioned more than once in the retrieved documents. Structured retrieval misses the concept of item identity other than defined by the location in the document structure. The above query retrieves documents about the Spanish painter and propagates the relevance scores down to the included entity mentions. A score aggregation per entity does not take place. XQuery principally enables the user to formulate the wanted score aggregation explicitly, but such queries get complex, require from the user to define the aggregation method manually, and are often inefficiently executed when the underlying join schema is not recognized. The following example shows such a query, where the user has chosen to sum up the document scores for each mention of an entity:

```
let $qid := tijah:queryall-id("//DOC[about(., Pablo Picasso)]")
let $mentions := tijah:nodes($qid)//entity[@type = "location"]
for $eid in distinct-values($mentions//@id)
    let $scores := for $mention in $mentions
    where $mention/@id = $eid
    return tijah:score($qid, exactly-one($mention/ancestor::DOC))
return <entity id="{$eid}" score="{sum($scores)}"/>
```

Notice that the up and down propagation of scores in structured retrieval performs a quite similar type of operation as required for entity retrieval and explicitly formulated in the above query. In this sense, structured retrieval only misses the necessary language concepts to be used for entity retrieval.

Entity Retrieval vs. Question Answering As mentioned in the introduction (see Sec. 1.1), entity retrieval and question answering have several overlapping interests. We consider here the tasks used in the corresponding TREC

evaluation as typical for question answering (Voorhees and Dang, 2005; Dang et al., 2006). A wide range of these assignments request the system to return so-called *factoids*. According to Voorhees and Dang (2005) factoid queries are asking for fact-based short answers. Though factoids are not clearly defined as "pure" entities, they contain entities in most cases. The track task even specifies the target type as either `person`, `organization`, or `thing`. Furthermore, question answering is not always about finding a single best matching answer. The TREC evaluation also knows "list" questions that ask for several instances of a given target type. Though the terminology differs slightly, this last described category of question answering tasks exactly meets the goal of entity ranking.

In general, question answering spans a wider range of issues and has a different focus than what we describe here as the entity ranking problem. Question answering starts already with the analysis of the (natural language) question, where the target answer type needs to be derived first (see e.g. Schlobach et al., 2007). Moreover, it is concerned with the extraction of appropriate answers from the text content of highly ranked passages or documents (see e.g. Lin, 2007). Both problems are lying outside the scope of the entity ranking task, since entity occurrences are recognized and tagged beforehand and requests on the answer type are assumed to be given explicitly. When studying entity ranking, the focus will lie on deriving entity relevance from the estimated relevance of text fragments, that contain occurrences of the entities. The problem can be regarded as a subtask of question answering, but seems not to be in the center of focus of the research in this field.

Different Tasks We can distinguish a few slightly different entity retrieval tasks, depending on how the search topic is expressed and which entity types are requested:

(1) creating a mixed-typed topic overview,
(2) type-dependent search of entities,
(3) completing a list of entities.

The first case is the most open task where the user only specifies the topic by a free keyword query. It will often be used for creating general topic overviews. Entities of all types are of interest here, but the user might appreciate if the system can detect and favor the most interesting types for a given topic. In case (2) not only the topic, but also the type is explicitly specified. We regard this as the most common entity retrieval task. List completion (3) refers to the case where the user knows a few entities fitting

topic and type and wants to find more of those. The assumption here is, that the user finds it easier to specify a few known relevant instances than to describe the topic of his/her interest and the type of entities he/she is looking for. To answer list completion queries a system has to derive the searched entity type automatically and has to find means to estimate the topical closeness of entities.

We find entity ranking also in specialized domains such as expert finding or timeline generation. In the first case entities of type `expert` have to be ranked according to their expertise on a given topic. The second example requires date entities to be ranked by their importance with respect to the topic of the timeline.

Supporting Text Since retrieval systems can only estimate relevance, users always have to verify whether returned results are indeed relevant answers to their request. Although entity ranking frees the user from the extra work of extracting the needed information from relevant text fragments, the "pure" entity names are often not enough to verify the relevance of a returned item. In this case supporting text snippets have to be found. However, we do not want to present all retrieved texts to the user – staying in direct conflict with the idea of entity retrieval – but the most supporting sentences only.

The problem of finding appropriate support sentences can be regarded as a task of its own, but it is highly linked to the problem of entity ranking. The collocation of the entity with topical relevant terms or other relevant entities might be a good indication that a text fragment supports the relevance of the entity.

4.2 Ranking Approaches for Entities

Whereas document, passage or XML retrieval employ standard retrieval models – passages or XML elements are regarded as small documents in that case – the same models would fail for entity ranking. The simple reason is that query words in general do not occur as a part of a named entity. Therefore, entity ranking is always based on the association between entities and documents. In general we will speak of text fragments instead of documents to capture also approaches that perform sentence or text window based entity ranking. Those text fragments are ranked first according to the query topic and their estimated relevance is propagated in a suitable way to the included entities of interest. This relevance propagation step will be the main issue for the rest of this chapter.

Related Approaches in Question Answering and Expert Finding As mentioned above, entity ranking is part of several question answering tasks and is also applied in the special domain of expert finding. We will thus take a look at related work first with the focus on the involved ranking of entities.

Many ranking methods in question answering are surprisingly unaware of the identity of answer candidates. We find query similarity scores of documents, passages, and sentences used as ranking features as well as linguistic part-of-speech patterns to filter out the most promising answer candidates (see e.g. Radev et al., 2002), but other mentions of the same answer do not influence its relevance. However, there are also methods that rank answers purely on their redundancy in a given set of relevant text fragments (Dumais et al., 2002; Clarke et al., 2001). Hence, they rank entities directly by their number of mentions. Others sum up the relevance scores of text fragments that contain string-identical answer candidates (Lin, 2007). Another recent study addresses the issue of answer identity in more detail by incorporating similarity scores between candidate answers in the calculation of the individual answer scores. Ko et al. (2007) suggest to employ a graphical model which effectively boosts candidate answers having many similar mentions, but at the same time avoids similar answers in the returned ranked set. The approach is thus aware of answer identity and tries to avoid duplicate mentions of identical answers in the result list, but it still does not rank entities but answer candidates.

Expert finding is an even younger field of information retrieval research. It has become popular after the upcoming of TREC's enterprise search task (Craswell et al., 2005). Early approaches build query-independent profiles for each candidate expert by merging all documents related to the candidate into one expert model. Experts are ranked then by measuring the similarity of their profile to the query (Liu et al., 2005). However, the most effective approaches on the TREC task measure instead the similarity between query and documents, and infer thereafter an expert ranking from the top retrieved documents. The former type of approaches is called *candidate-centric*, the latter *document-centric* (Balog et al., 2006). Also combinations of both approaches are existing (Serdyukov and Hiemstra, 2008).

When inferring expert ranks from related documents in the document-centric approach, we see again different strategies used. Algorithms of the one kind rank candidates by the aggregated relevance of all related top documents (Balog et al., 2006; Macdonald and Ounis, 2006). Other proposed methods build query dependent social networks from the top retrieved documents (Campbell et al., 2003; Chen et al., 2006). More precisely, so-called bibliographic coupling graphs are generated by modeling related documents as links between persons (for instance by utilizing the `from` and `to` fields

in emails). Candidates are ranked then on such social networks by popular *centrality measures*, such as Kleinbergs HITS algorithm (Kleinberg, 1998). However, these centrality based approaches have failed so far to show similar performance as the simpler aggregation methods. Both aggregated relevance and centrality based methods still ignore some properties of the data. Methods using aggregated relevance do not reflect the relation between experts, whereas the centrality measures on the coupling graph simply model documents as unweighted links between candidates, neglecting their relevance to the query. We will show later in this chapter that graph-based approaches are able to incorporate both kinds of information.

Learning from Graph-Based Retrieval We have seen in related work on expert finding that graphs are used to model the relation between entities. Although the existing graph models for expert finding still fail to model all available information, they show several advantages compared to non-graph-based approaches. Firstly, graphs make the propagation process more transparent. It becomes easy to describe and to visualize. Secondly, graphs allow to discover and use indirect connections. They show relations between entities that are never mentioned together, but often in the neighborhood of a common third entity. Finally, we show that even non-graph-based approaches for entity ranking can often be interpreted in terms of a graph-based equivalent.

Graph-based ranking methods are first of all known from web retrieval. Among them, Pagerank (Page et al., 1999) and HITS (Kleinberg, 1998) are probably the most popular, and their usage is widely studied in the field of hypertext retrieval. Similar to our work here, more recent graph-based approaches try to incorporate as much information into the graph as possible. Pagerank can be regarded as a Markov process, or a random walk on the web graph (Henzinger et al., 1999). Several attempts have been made in the last years to make such a walk query and content dependent. The *intelligent surfer* walks to linked pages biased by their relevance to the query (Richardson and Domingos, 2001). The surfer model proposed by Shakery and Zhai (2006) uses a similar, but bi-directional walk considering both out-links and in-links of a node.

Graph-based ranking methods often find applications beyond the bounds of hyperlinked corpora. We can find them used among others for spam detection (Chirita et al., 2005) and blog search (Kritikopoulos et al., 2006). Kurland and Lee (2005, 2006) experimented with structural re-ranking for ad-hoc retrieval, first using Pagerank and later HITS in bipartite graphs of documents and topical clusters. Erkan and Radev (2004) used implicit links

between similar sentences to compute their centrality for text summarization. More close to our work, Zhang et al. (2007) studied query-independent link analysis in post-reply networks for expert detection comparing Pagerank and HITS centralities.

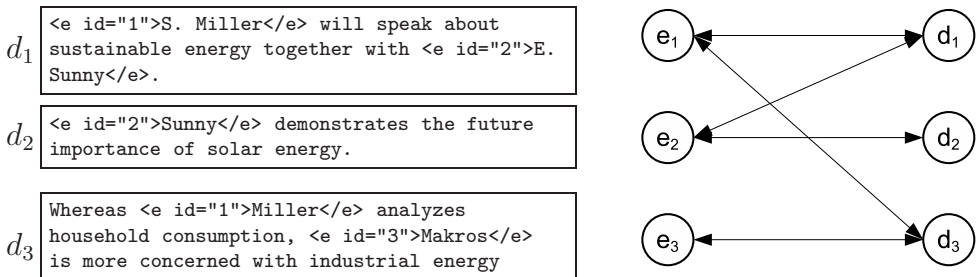
The various applications of graph-based ranking strategies show that graph-based centrality features are not only suitable for ranking in hyperlinked web corpora, but can be applied in a more generic way to ranking problems where links between items mean that they support each others relevance. Agarwal et al. (2006) and Chakrabarti (2007) show such a generic framework for graph-based entity ranking in their recent work. They generalize completely from the application and even use a broader notion of entities which includes the documents themselves. Agarwal et al. examine ways to learn an edge weighting function for a Markov walk from relevance assessments, while Chakrabarti focuses mainly on performance issues for the computation of personalized Pagerank vectors. Both studies still do not test the precision on entity ranking tasks paired with user relevance judgments.

Processing Model Graph-based entity retrieval includes the following processing steps. While the named entity recognition can take place beforehand, the query dependent processing is divided in three consecutive steps:

(1) Initial retrieval and scoring of text fragments,
(2) Building of an entity containment graph,
(3) Relevance propagation within the graph.
(4) Filtering out entities of the requested answer type.

The first step remains a standard retrieval task on the entire text collection, which selects the most relevant text fragments according to the query topic. Those are used in the second step for building a graph that models the containment of entities within retrieved text fragments (Section 4.3). In the third step we exploit the graph structure in order to rank the entities, respectively, to propagate the relevance information (Section 4.4) within the graph. Finally, the result has to be filtered if the query asked for a specific entity type, but other entities were included in the entity containment graph as well.

The first step allows to apply any kind of retrieval model known from document retrieval, but some of the later relevance propagation models require all scores to be probabilistic. The size of the requested text fragments – entire documents, passages, or sentences – remains an interesting parameter for testing. List completion tasks require a different processing, starting from a seed set of entities rather than from an initial query. However the



(a) Tagged text fragments (b) Corresponding graph

Figure 4.1: Expertise Graphs

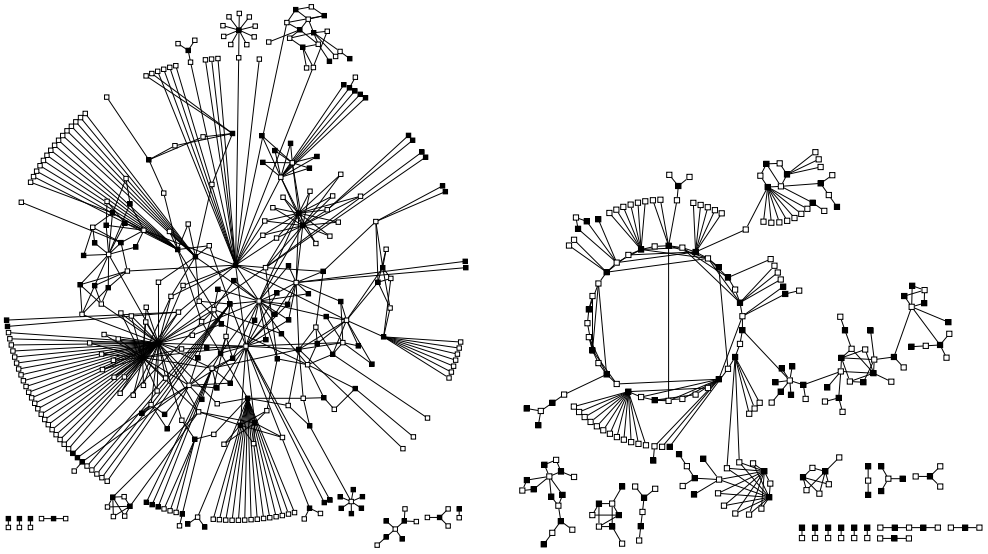
specialized task will not be discussed further or analyses in this study.

4.3 Entity Containment Graphs

This section proposes and discusses the modeling of appropriate graphs that represent the association between entities and documents. We will further on call them *entity containment graphs*.

Suppose we have a set of documents D, in general *text fragments*, with relevance scores from an initial retrieval run and a set of *entities* E, that finally should get ranked according to the given query q. Furthermore, we know the containment relation of text fragments and entities, i.e. for each text fragment which entities are occurring inside. This relation can be represented in a graph, where both text fragments and entities become vertices and directed edges symbolize the containment condition. Such a basic entity containment graph is always bipartite, since all edges connect text fragments with entity vertices (see Figure 4.1). Entity containment graphs have been used also for co-ranking of authors and documents (Zhou et al., 2007).

Figure 4.2 shows typical entity containment graphs computed for two expert search queries from the TREC 2007 enterprise track. As we see in the example figures, entity containment graphs often consist of one main component connecting most of the vertices and several smaller other ones. The size and number of these smaller components changes from query to query. The graph representation provides several useful features of the included entities. It shows in how many different text fragments they are occurring, and also, whether they are connected over common text fragments with other entities, or remain uncoupled (like vertices in the lower part of the figure). Behind the last feature stays the hypothesis that entities mentioned in the same text fragment also have a stronger relation to each other than those which never



(a) CE-002: "hairpin RNAi gene silencing" (b) CE-006: "sustainable ecosystems"

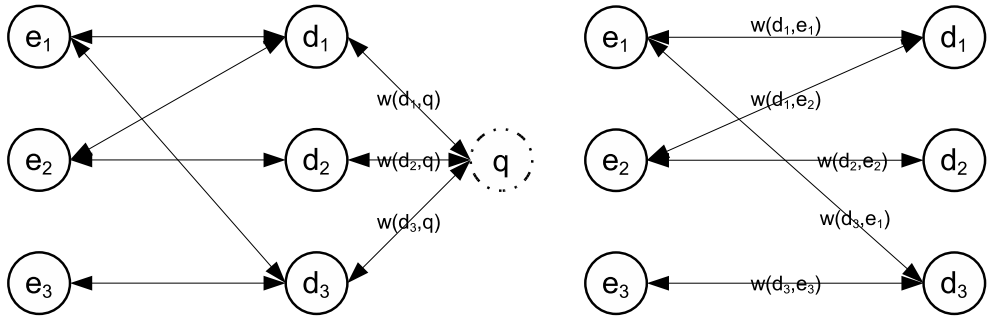
Figure 4.2: Entity containment graphs computed for TREC expert search topics, white vertices mark entities, black documents

appear together. Notice that in contrast to the bibliographic coupling graph, which models documents as edges between entities, such a bipartite graph of text and entity vertices captures both the direct containment relation as well as the indirect 2nd-degree (in general n-degree) neighborhood of entities to each other.

4.3.1 Modeling Options

In the following we show several modeling options and parameters to further improve the density of the graph. Since we are interested in the propagation of relevance through the graph network, it is important to exploit all known connections between the entities of the graph.

Modeling Text Fragment Scores The initial retrieval run does not only return a ranked list of text fragments, but also their corresponding relevance score according to the given query. A simple way to incorporate such prior knowledge into the graph model is to add a further "virtual" query vertex q to the graph, which is connected to all text fragments. We can then define an edge weight function $w(d, q)$ which assigns probabilities to all new edges



(a) Query Weights (b) Association Weights

Figure 4.3: Modeling query and association weights

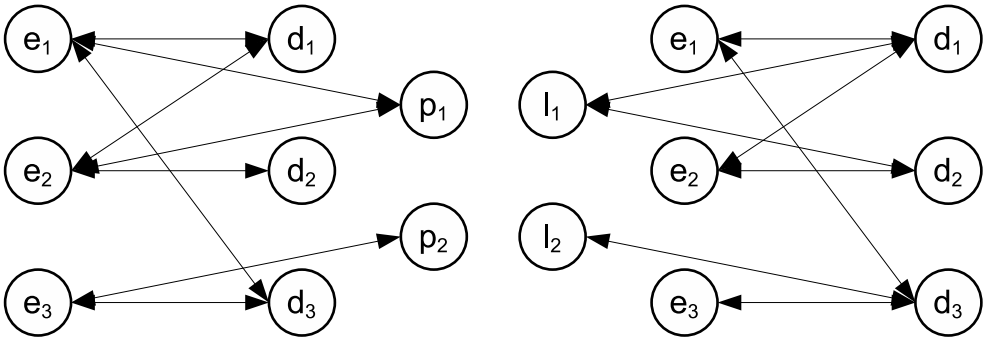
(see Figure 4.3(a)):

$$w(d,q) = P(d|q) = \frac{score(d|q)}{\sum_{d'} score(d'|q)}.$$

The additional query node is represented here as an additional "entity" contained in all documents. In order to motivate this modeling, one should think of the query as a set of terms, which is indeed contained in those documents to a certain degree, corresponding to the initial score.

Association Weights between Documents and Entities The graph contains a directed edge from the document to each included entity. However, it does not provide so far any information about the strength of this association, in other words, how important the entity is for the document. Another edge weight function $w(d,e)$ can be defined to capture this information (see Figure 4.3(b)). Without any further domain knowledge, all occurrences of an entity should be treated equally. In a better known domain, like the expert finding task, occurrences of an expert in a document might be weighted differently. If an expert is the author of an email, she/he is probably more influential on the content than another expert who is just mentioned somewhere in the text.

Modeling Overlapping Text Fragments Choosing the right text fragment size is not an easy decision. Smaller text windows provide stronger evidence of the semantic connection of their contained entities and also ensure a clear connection to the query topic. Larger text windows, on the other hand, come with a higher chance to find entity cooccurrences, which are the basis of the later described relevance propagation step. Related work on expert finding



(a) Further text fragments (b) Further entity types

Figure 4.4: Modeling additional information

shows that proximity features help to further improve the retrieval quality. Proximity features have been integrated either in the relevance estimation model itself (Petkova and Croft, 2007), or by tightening the initially ranked text fragments (Zhu et al., 2006). A graph model is able to include both, the small sized paragraphs and the larger documents (see Figure 4.4(a)). This way it keeps the connectivity of the larger sized text fragments, but emphasizes the connections of higher evidence. We have shown in the figure only edges from documents or paragraphs to entities. One could also include edges between documents to paragraphs to visualize the containment relation. Such expansion, however, would break with the bipartite property of the graph.

Including Further Entity Types For typed entity ranking tasks (see Section 4.1 task type 2) the focus of interest will lie on a certain *entity type*. Although a task like expert finding is only concerned with retrieving expert entities, it might still be useful to include nodes of other entity types into the graph (see Figure 4.4(b)). The motivation behind such a graph expansion would be to show the connection between entities of different types and to increase the relevance propagation between them. If one would search for instance for important dates in the life of the painter Pablo Picasso, it is probably useful to add more than date entities to the graph. In this case, further person or location entities might reveal important connections as well.

Including Further Edges The suggested entity containment graph only models the relation of documents and included entities. One modification could be to include further document to document or entity to entity edges

(see Figure 4.5). The first ones for links between documents, the second ones if the found entities are standing in a known relation to each other. We think here for instance of exploiting known hierarchical ontologies, like *Cape Town* is part of *South Africa*, or *21 April 2006* and *2006* are date entities supporting each other. In case of expert finding, enterprises will often have a hierarchical organization overview of its personnel. By including such additional edges, the graph gains a higher density and enables more relevance propagation, but it looses its strict bipartite property.

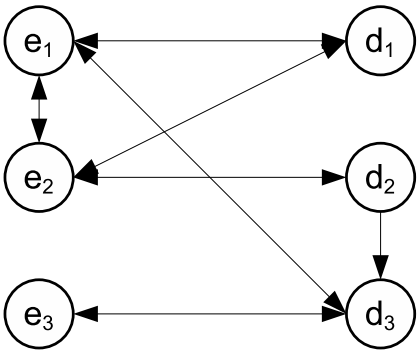


Figure 4.5: Modeling direct connections between entities or documents

Controlling Graph Size and Topical Focus Apart from the graph modeling itself, the most influential parameter on the graph size and density is the number of top ranked documents taken into account while building the graph. Notice that for the unweighted graph only the restriction to the top ranked documents makes the graph model query dependent. Hence, by including more lower ranked documents more included entities are found and usually the graph's density increases with the drawback of loosing the topical focus of the graph.

4.4 Relevance Propagation

Once having an entity containment graph, there are several relevance propagation models that can be used for the ranking of entity vertices. The underlying assumption is that the connectivity of an entity within the graph shows its importance to the given query topic. Graph theory has developed the concept of *centrality*. It is defined there as a *structural index* (Koschützki et al., 2004a), with the implication that vertices are assigned values according to their structural importance and structural equivalent nodes are assigned the same centrality value. Notice that a structural index in graph theory is completely different from structural indices as discussed in the previous chapter. Various centrality indices have been designed in the last decade using vertex features like degree, or distance in the graph, or even recursively defined features as the centrality of neighboring nodes. Graph-based ranking approaches often make use of such centralities, especially the most known ones, HITS (Kleinberg, 1998) and Pagerank (Page et al., 1999), are applied

to many different tasks (see Section 4.2). Instead of using "pure" structural graph features, node weights of prior document retrieval are incorporated to achieve query dependent centrality measures. However, such a centrality concept moves away from original definition being a solely structural index. We will therefore speak of *relevance propagation* in graphs, which describes the aim of distributing relevance from initial sources – the pre-scored text fragments in this case – throughout the graph network and especially towards the entities of interest. All propagation methods introduced in this section incorporate the initial query scores. An *unweighted* counterpart can always be obtained by simply setting all weights to 1. We will later compare the retrieval performance of the unweighted variants, depending purely on the structure of the graph with the weighted models that propagate the initial document weights through the graph network.

For abbreviation, the set $\Gamma(v)$ denotes all vertices adjacent to vertex v. Furthermore, we use different letters to distinguish between a document vertex d and an entity vertex e.

Relevance of Text Fragments Before starting a graph-based relevance propagation, we need to score all vertices in the containment graph whose relevance can be estimated directly, namely all text fragments. Since some of the propagation models that will be introduced in this section follow a probabilistic approach, it is necessary to employ a probabilistic score function.

Unless mentioned otherwise, we determined the relevance of text fragments following the language modeling approach (Hiemstra and Kraaij, 1998; Miller et al., 1999). A simple Jelinek-Mercer smoothed scoring estimates the probability of the query q generated by a given text fragment d within the collection C:

$$P(q|d) = \prod_{t \in q} (1 - \lambda) P(t|d) + \lambda P(t|C).$$

Compared to the *NLLR* (see Section 2.2) which is used in many other cases throughout this thesis, the above equation does not produce scores in log-space.

Following the Bayes' theorem, $P(d|q) = P(q|d)P(d)/P(q)$, the probability $P(q|d)$ is turned into a relevance estimation of the text fragment d. When building the entity containment graph from the top ranked text fragments only, we assume that the probability of a user visiting a lower ranked text fragment equals zero. Consequently, we normalize the probability distribution within the set of top ranked text fragments R. The prior probability $P(d)$ is assumed to be distributed uniformly, since we have no other evidence

of relevance in this case than the query itself:

$$P(d|q) = \frac{P(q|d)P(d)/P(q)}{\sum_{d' \in R} P(q|d')P(d')/P(q)} = \frac{P(q|d)}{\sum_{d' \in R} P(q|d')}.$$

For comparable results, the weighting function used in non-probabilistic propagation models is set accordingly: $w(d, q) = P(d|q)$.

Association Weights vs. Transition Probabilities As explained in the previous section, association weights between entities and text fragments are set uniformly to 1, if we do not have any domain knowledge that gives reason to prefer certain connections over others. Probabilistic propagation models, however, require edge transition probabilities instead of association weights. Such transition probabilities $P(d|e)$ and $P(e|d)$ can be derived by normalizing the association weights of outgoing edges (corresponding to Balog et al., 2006):

$$\begin{aligned} P(e|d) &= w(d, e) / \sum_{e' \in \Gamma(d)} w(d, e'), \\ P(d|e) &= w(d, e) / \sum_{d' \in \Gamma(e)} w(d', e). \end{aligned}$$

Notice that the probabilistic transition model looses the symmetry of the edge weighting function: $P(e|d) \neq P(d|e)$. A vertex with a high number of outgoing edges assigns lower transition probabilities to each single outgoing edge than another vertex having less outgoing edges. A uniform edge weighting is therefore not leading to uniform transition probabilities.

4.4.1 One-Step Propagation

One-step propagation models correspond to degree centralities that only take into account directly adjacent vertices.

Maximal Retrieval Score The simplest model of entity ranking can be described by the following process. Walking down the ranked list of documents, we add all included entities that have not been encountered before in that order to the final ranked list. The equivalent propagation model on the entity containment graph assigns to each entity vertex the weight of the highest ranked linked document node:

$$wMAX(e) = \max_{d \in \Gamma(e)} w(d, e)\ w(d, q).$$

We also take the association weights $w(d, e)$ between documents and entities into account. If the weights are set uniformly to 1, the equation indeed models the described simple selection process. Although the model is formalized within the graph-based framework, it ignores most of the features provided by the entity containment graph. We will refer to it later as a *baseline* ranking model in order to compare it to other relevance propagation models that consider more graph features.

Weighted Indegree When the maximum in the above propagation model is replaced by the sum of adjacent vertices, the model rewards often occurring entities:

$$wIDG(e) = \sum_{d \in \Gamma(e)} w(d, e)\ w(d, q).$$

We name this propagation model *weighted indegree* $wIDG(e)$, since it corresponds in its unweighted version with an indegree centrality.

The propagation model corresponds to other approaches not explained in terms of graphs. The theoretically most sound methods for expert finding proposed by Balog et al. (2006) and Macdonald and Ounis (2006) can be expressed as an expertise inference on a linear Bayesian network $q \rightarrow d \rightarrow e$:

$$P(e|q) = \sum_{d \in D} P(e|d)\ P(d|q).$$

It uses the query to find relevant documents and then candidate experts occurring in these documents. The higher the number of relevant documents mentioning a candidate expert, the higher its probability of being an expert. The initial document scores are aggregated with respect to the related candidates. It is easy to see that this model is equivalent to the above introduced weighted indegree $wIDG(e)$, if weights are distributed as probabilities. It is important in this case to point out the consequences of using a probabilistic model, compared to association weights $w(d, e)$ that do not have to satisfy the condition $\sum_{e'} w(d, e') = 1$. A highly relevant text fragment containing many entities has to distribute its relevance in the Bayesian network over all entities. Hence, its relevance contribution to the individual entity might remain small. When using a uniform edge weight function in the weighted indegree model, which sets all $w(d, q)$ to 1, the same highly relevant text fragment propagates its relevance undivided to all included entities.

4.4.2 Multi-Step Propagation

One-step propagation models calculate the relevance of an entity by looking only at the query dependent relevance of the text fragments it occurs in.

Other features, like the co-occurrence with other entities are not taken into account so far.

If a user manually tries to find and rank entities, the process would probably look different than the one described by the one-step propagation models. Starting with a relevant document she might discover a few entity candidates. In order to estimate their relevance she might look up in which other documents the entities are mentioned and furthermore which other entities are mentioned there as well. After a few steps from documents to entities and back, she will return to the initial retrieved list of documents and restart the process from another document.

Such a search behavior is modeled by random walk on the entity containment graph. The user "moves" constantly over the edges of the entity containment graph and we model the probability that the user is stationed at any time at a certain vertex. The stationary probabilities of an infinite random walk, would be completely independent of the initial vertex weighting. However, a slightly changed random walk model enables to retain the influence of the initial query weighting also for infinite processes. It is just necessary to consider that the modeled user will once in a while return to the initial retrieved list of documents to ensure that the found entities are close enough to the topic of interest. Such a behavior can be modeled by introducing the possibility of a weighted random jump to the walk model. The jump allows to reach a document vertex from any other vertex in the graph. Since it is more likely that a user returns to a higher ranked document, we bias the probability to reach a document vertex via random jump by its initial probability of relevance. Such a query-dependent random walk model is also called a *personalized* centrality index (Koschützki et al., 2004b):

$$\begin{aligned} P(e) &= \sum_{d \in \Gamma(e)} P(e|d)P(d), \\ P(d) &= \lambda P(d|q) + (1-\lambda) \sum_{e \in \Gamma(d)} P(d|e)P(e). \end{aligned}$$

The damping factor λ specifies the probability of either following an edge of the graph or jumping randomly to one of the retrieved documents. Our model does not contain a probability to jump to an arbitrary entity at any time. A random picking of an entity is not clearly motivated by the underlying process model of user searching for entities. Since we have no prior evidence of the entities' relevance, such a jump towards entities would be unweighted, thus having uniform probabilities to reach any entity. Notice also that the random jump towards documents already allows to reach all components of the graph and thus indirectly connects all entities. Therefore, we abandoned

a random jump towards entities.

In order to compute the stationary probabilities of the above defined walk, a step-wise calculation of probabilities is executed until the values converge. In general, walks – especially on bipartite graphs – are not guaranteed to converge. If we consider for instance a walk without jumps that initially assign zero probabilities to all entity vertices, we can observe that such a model alternately assigns zero probabilities to either documents or entities and never converges. Our infinite walk model, however, overcomes the problem by the introduction of the random jump. It allows to reach document vertices in any step and therefore opens up the strict mutual walk model from documents to entities and back.

Despite of the distinction between entity and document vertices, the described infinite walk model is equivalent to the definition of the *personalized* Pagerank (Page et al., 1999).

Comparison with HITS The distinction of the vertices into two classes – text fragments and entities – reminds of the HITS algorithm. HITS was originally used to characterize a hyperlinked network of web-pages consisting of portal pages with a high number of outgoing links, so-called *hubs*, on the one hand, and cited content bearing pages with a higher number of in-links on the other hand, so-called *authorities*. HITS does not require two distinct sets of vertices as in our case, but classifies the vertices itself according to their properties. It is based on directed graphs in contrast to our undirected entity containment graph. However, we can easily transfer the algorithm to our bipartite entity containment graphs with text fragments (*hubs*) pointing to entities (*authorities*). Instead of directing all edges from documents towards entities, we define equivalently that documents get *hub* scores and entities get *authority* scores only. Furthermore, also HITS can get "personalized" by introducing a weighted random jump:

$$\begin{aligned} Auth(e) &= \sum_{d \in \Gamma(e)} w(e,d)\ Hub(d), \\ Hub(d) &= \lambda\, w(d,q) + (1-\lambda) \sum_{e \in \Gamma(d)} w(e,d)\ Auth(e). \end{aligned}$$

The here presented model resembles the randomized HITS of Ng et al. (2001), apart from limiting the random jump to reach document vertices only. The HITS algorithm was adapted in other ways to incorporate a prior vertex weighting, among others by Bharat and Henzinger (1998). Since we use HITS only for comparison, we tried to model it as similar to the before defined infinite walk as possible.

Algorithm 3: Iterative HITS Algorithm

```
HITS(Graph G(V, E)) ≡
  BEGIN
    initialize hub(v) and auth(v) for all v ∈ V;
    WHILE hubs and authorities not converged DO
      calculate auth(v) for all v ∈ V;
      calculate hub(v) for all v ∈ V;
      normalize authorities;
      normalize hubs;
    RETURN authorities, hubs
  END
```

In fact, the only remaining difference between the above personalized HITS definition and the before shown infinite random walk concerns the edge transition weights. Notice that the HITS algorithms (see Algorithm 3) performs a normalization step on its *hub* and *authority* values after each iteration. Therefore it does not need to have real transition probabilities. We can for instance set them uniformly to 1 with the implication that a document does not divide its importance among its contained entities but propagates its full weight to all of them, vice versa for the propagation from entities to documents.

In this respect, HITS can be seen as a straightforward extension of the weighted indegree model, which could also use edge weights instead of transition probabilities.

4.5 Experimental Study I: Expert Finding

A quite typical example of an entity ranking task is the problem of *expert finding*. In expert finding, as performed in TREC's enterprise track (Craswell et al., 2005), a system has to come up with a ranked list of expert entities with respect to a given topic of expertise, a corpus of enterprise documents, and a list of the employees of the company as candidate entities. We therefore used TREC's expert finding task for a first evaluation of our entity ranking approach. TREC changed the corpus and in a more subtile way also the task itself between 2006 and 2007. We will report here results of both years.

Expert Finding Task 2006, W3C Corpus The corpus used in that year was the *W3C-corpus* representing the internal documentation of the World Wide Web Consortium (W3C). It was crawled from the public W3C (*.w3.org) sites in June 2004. The data consists of several sub-collections: web pages,

source code, and mailing list archives. All experiments in this section are performed on the email part of the W3C corpus (1.85 GB, 198,000 emails), which is the most clean and structured part of the corpus. Using the entire W3C corpus yielded in slightly worse results in general, however the order of compared techniques with respect to their retrieval quality remains the same. A list of 1092 potential experts with full name and email, all of them participating in the W3C working groups, is provided with the TREC data. We preprocessed the corpus data in order to convert it into well-formed XML with the least possible changes to the data itself, and secondly for tagging all occurrences of experts within the corpus. A simple string-matching tagger marked a candidate when it either matched the complete candidate name or her/his email-address. We disregarded abbreviations of the names since they could also mislead to different persons. Other occurrences of email addresses were tagged as well as non-candidate persons.

Expert Finding Task 2007, CSIRO Corpus The data used in TREC 2007 is a crawl from publicly available pages of Australia's national science agency CSIRO. It includes about 370,000 web documents (4 GB) of various types: personal home pages, announcements of books and presentations, press releases, publications. The task itself also changed slightly since no list of candidate expert was provided as in 2006. Instead only the structure of candidates' email addresses was given: *firstname.lastname@csiro.au*, and systems had to recognize experts themselves using this pattern. We built therefore in a first step our own candidate list as well as a list of other persons, searching for the provided pattern, respectively a general pattern of email addresses for non-candidate persons. About 3500 candidate experts and around 5000 others persons were identified this way by full name and email address. In a second preprocessing step all mentions of either full name or email address were tagged in the corpus data assigning unique person or candidate identifiers.

Ranking and Graph Building The initial ranking as well as the graph generation was expressed by XQuery statements and executed with PF/Tijah. For this experiment, we generated XQueries that directly output entity containment graphs in *graphml* format[2] given a title-only TREC query. A standard language modeling retrieval model with Jelinek-Mercer smoothing was employed for the initial scoring of text nodes (see Section 4.4). The generated graphs were later analyzed with a Java graph library, adapted by our own weighted propagation models.

[2] `http://graphml.graphdrawing.org`

The HTML structure of the CSIRO corpus allowed to include text fragments of different size into the entity containment graph. We experimented on this corpus with building entity containment graphs containing either only document vertices or document and paragraph vertices together. In the second case, we performed a second paragraph ranking of those paragraphs included in the top retrieved documents of the first document ranking.

4.5.1 Result Discussion

The result analysis is based on standard retrieval quality measures such as mean average precision (MAP) and precision at top 5 ranked experts (P@5). P@5 is used here instead of the otherwise reported P@10, since the expert ranking task on the CSIRO corpus has a very low number of relevant entities, on average 3 per topic. Therefore we even observe P@5 values lying below the measured MAP for most experiments on this track.

Although it is objectionable in general to set parameters differently for different test collections, since it enables overfitting, we have made one exception. The number of top ranked documents that was used to build the entity containment graph needed to be analyzed in a quite different value range for the two test collections. Figure 4.9 and Figure 4.10, which will be discussed later, show clearly the diverse behavior. We assume that the effect is caused by two factors. Firstly, the TREC 2006 topics know a clearly higher number of relevant experts compared to the judgments of 2007. Hence, for the 2006 topic set including more documents in the evaluation increases the chance to deliver more relevant experts, whereas the 3 relevant experts of the 2007 topics are often found in the first few documents. Secondly, the effect might also be caused by the different characteristics of the two text collections. The email data of the W3C corpus is in general more repetitive than the web pages of CSIRO. Often a few web-pages of a certain research group contain all relevant experts and the likelihood to relevant experts anywhere else remains extremely small. In order to find a comparable parameter setting of the two collections, we set the number of included documents not to the value yielding the highest precision, but to the point where recall is not increasing significantly any further. Therefore, all later mentioned experiments on the W3C corpus use the top 1500 retrieved documents, whereas for the CSIRO corpus only the top 200 documents are included in the graph.

We start the evaluation by looking at the introduced propagation methods and their parameters. The influences of modeling options of the entity containment graph will be presented and discussed thereafter.

	W3C corpus		CSIRO corpus	
	MAP	P@5	MAP	P@5
IDG	0.326	0.555	0.288	0.168
wMAX	0.355	0.596	0.313	0.176
wIDG	0.375	0.629	0.351	0.208
wIDG-norm	0.372	0.633	0.330	0.212

Table 4.1: Retrieval quality of one-step propagation methods

One-Step Propagation Models The set of one-step propagation models is extended for the experiments by 2 further variants of the indegree model. The unweighted indegree *IDG* represents a pure centrality score depending solely on the structural properties of a vertex. Moreover, the unweighted indegree is equivalent to the so-called voting model often applied in question answering (Lin, 2007) and also in expert finding (Macdonald and Ounis, 2006). The degree of an entity vertex is equal to the number of votes it gets by mentions from document vertices. The other additional indegree variant shows the effect of normalizing all edge transition probabilities. It represents a fully probabilistic propagation model identical to the one proposed by Balog et al. (2006).

Table 4.1 gives a result overview of the one-step propagation methods. The results of the unweighted indegree methods first of all shows that graph features are meaningful for entity ranking. Even without information about the documents content, the indegree enables to rank entities appropriately. However, the pure centrality based entity ranking still stays behind the simple maximum retrieval score approach *wMAX*, which shows that the initial document ranking carries such meaningful relevance evidence that every successful method needs to incorporate that information. Consequently, the weighted indegree *wIDG*, using both graph features and the initial scoring, easily outperforms the other techniques.

Another important observation concerns the normalization of the edge transition probabilities. We can see that edge weight normalization drops the performance slightly on the W3C corpus, and even considerably on the CSIRO corpus. The results indicate that the required edge weight normalization for probabilistic walks does not correctly model the actual relevance evidence an entity gets from its associated documents. It is apparently not appropriate to assume that an important document needs to share its influence among all mentioned entities, it should instead support each mentioned entity with its entire undivided relevance.

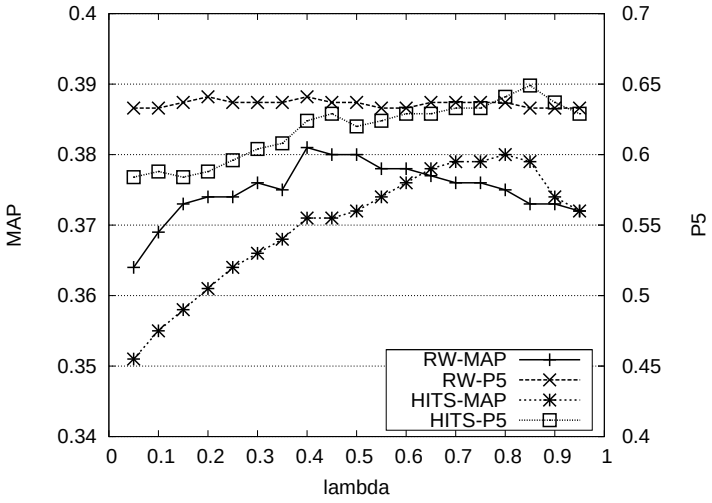


Figure 4.6: Adjusting the random jump probability, W3C corpus

Multi-Step Propagation Models In order to demonstrate the performance of the multi-step propagation models, it is important to first tune the steering parameters of the random walk. Since our collection data provides a poor base for connections among documents or entities themselves, we only analyze walks over document-entity edges combined with random jumps on the graph. Hence, the steering of the walk falls back to a single parameter λ, that determines the probability of a random jump.

Figure 4.6 and Figure 4.7 show that the best setting of λ differs between the common random walk model RW and the HITS-like variant. Following the MAP curves, we see in the figures of both collections that the common random walk shows its best retrieval quality for a smaller value of λ than the HITS model. The precision on top of the retrieved list P@5 seems in general less influenced by the choice of λ. Although the graphs in the two figures do not have the exactly same maxima, it is possible to set the random jump probability in a uniform way without sacrificing retrieval quality noticeably. The random walk λ is fixed to 0.4 for all following experiments whereas the HITS λ is set to 0.85. Such a uniform setting also prevents from over-fitting for a single collection.

Furthermore, the figure shows that the HITS variant of the random walk works as good as the other one on the W3C data and considerably better than the other on the CSIRO collection. The result directly corresponds to the before made observation concerning the effect of edge weight normalization. Since the HITS walk uses a different normalization technique, it is not influenced by the negative effect of edge weight normalization.

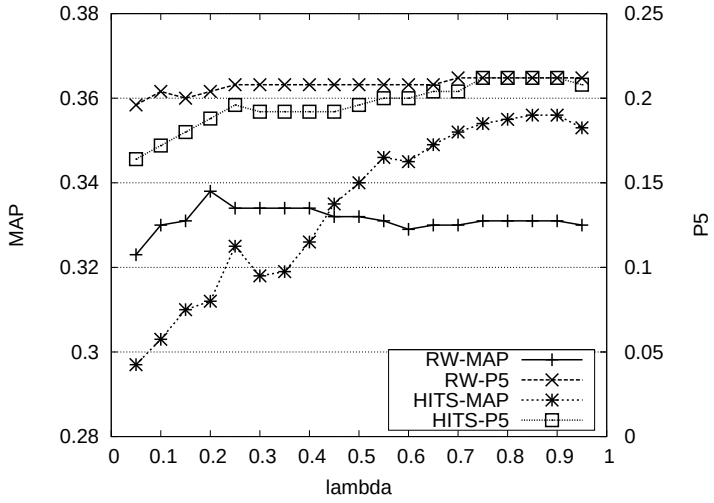


Figure 4.7: Adjusting the random jump probability, CSIRO corpus

The stationary probabilities of a random walk represent the chance that the "walker" is present at a certain node after an infinite number of steps in the graph. The iterative calculation of the stationary probabilities performs an n-step walk starting from an initial probability distribution that stops as soon as the values converge. If we start the walk with the initial distribution given by the query scores and stop the walk after a certain number of steps n, we get the results displayed in Figure 4.8. The figure shows the common random walk model only, not its HITS variant.

Whereas a converging walk requires between 50 to 100 iterations depending on the setting of the convergence test, we see in the figure that in fact a rather limited number of steps is sufficient to reach the maximal retrieval quality. A one-step walk in this model is equivalent to the normalized weighted indegree model. The figure thus also shows that multi-step propagation models can further improve the retrieval quality of the one-step propagation models.

Number of Top Retrieved Documents As we stated already before, the number of top retrieved documents k included in the graph modeling has an important impact on the retrieval performance and depends highly on the collection and expected number of relevant entities. Figure 4.9 and Figure 4.10 shows furthermore how the different propagation models are influenced by the number of included documents k. Obviously, the retrieval quality of the unweighted indegree drops when the graph looses its tight topical focus. When the number of documents increases, this ranking technique suffers from the

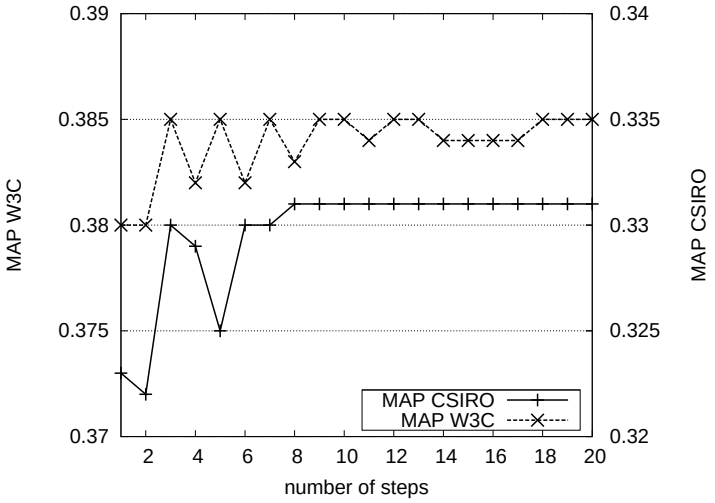


Figure 4.8: Retrieval quality vs. walk length

fact that it cannot distinguish the relevance of the included documents. All other ranking approaches that propagate the initial retrieval scores through the graph, are not affected by the decreasing topical focus of the graph. In case of the W3C corpus, the mean average precision is even constantly increasing over the displayed range of k for all 3 propagation methods, since the higher number of included documents slightly increases the recall.

Comparing the *wIDG*, common random walk, and HITS, the figure approves our previous results. The common random walk looses when edge weight normalizations shows a negative effect as on the W3C corpus. Otherwise all 3 approaches reach a similar retrieval quality with the multi-step random walks slightly on top of the indegree model.

Including Persons, Paragraphs, or Associations Weights In the following, we present the test results of several graph modeling options introduced in Section 4.3.1.

During corpus preprocessing, all mentions of expert candidates and other persons had been tagged, which could be identified by their email address. However, all experiments reported so far used only the candidate entities and did not include non-candidate persons into the entity containment graph. Our hypothesis was that random walk models might be able to profit from the additional connections emerging from the further person vertices in the graph. Table 4.2 does not show support for the hypothesis on both collections. Although the other persons do not harm the retrieval quality, their inclusion also shows no positive effect.

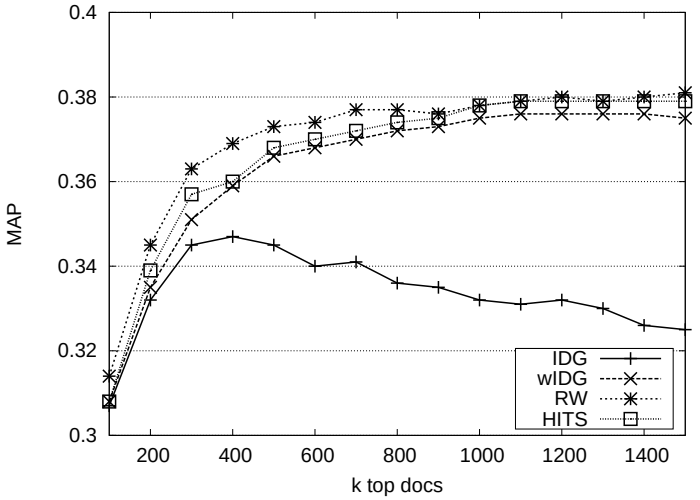


Figure 4.9: Influences of the number of included top retrieved documents k, W3C corpus

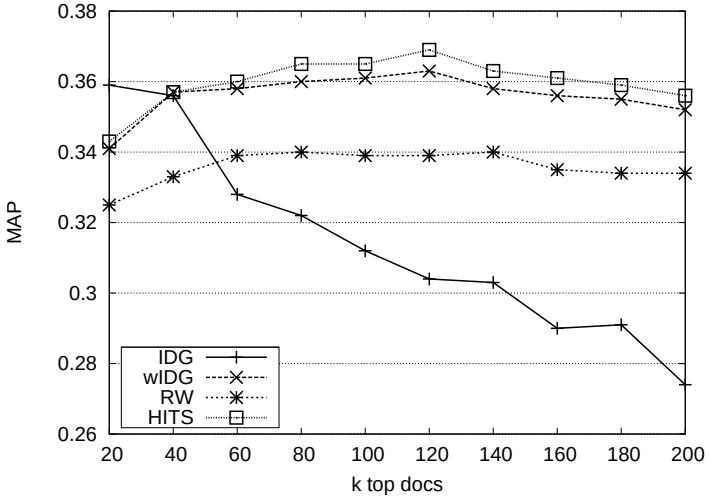


Figure 4.10: Influences of the number of included top retrieved documents k, CSIRO corpus

		W3C corpus		CSIRO corpus	
		MAP	P@5	MAP	P@5
normal	wIDG	0.375	0.629	0.351	0.208
	RW	0.381	0.641	0.334	0.208
	HITS	0.379	0.649	0.357	0.212
person	wIDG	0.375	0.629	0.351	0.208
	RW	0.376	0.645	0.335	0.208
	HITS	0.377	0.637	0.356	0.212
edge-weights	wIDG	0.373	0.616		
	RW	0.378	0.637		
	HITS	0.376	0.637		
paragraphs	wIDG			0.383	0.220
	RW			0.377	0.216
	HITS			0.389	0.220

Table 4.2: Comparing different graph modeling options

Another proposed modeling variant uses edge weights to model the association between documents and experts. In case of the W3C emails from 2006, we had the chance to experiment with such association weights. Instead of weighting all mentions of an expert equally, we assigned different weights to occurrences in the emails' `FROM`, `TO`, and `CC` fields or the text body. Instead of experimenting with all possible weight settings ourselves, a uniform weighting was compared to the setting as suggested by Balog and de Rijke (2006): $w(e, D_{from}) = 1.5$, $w(e, D_{to}) = 1.0$, $w(e, D_{cc}) = 2.5$ and $w(e, D_{body}) = 1.0$. If a person appeared in several fields, only the maximum of the association scores is considered. In case of the CSIRO data-set, the same distinction was not possible since the web documents do not contain elements that suggests a different association between document and candidate. Unfortunately, the outcome of this test neither confirmed the results of Balog and de Rijke, who tested themselves on the expert finding query set of TREC 2005, nor does it show any improvements at all compared to the uniform assignment of edge weights. We even tested a few further ad-hoc weighting schemes not reported here, but could not yield noticeable improvements over the uniform model.

As a last modeling option, we tested the inclusion of further smaller-sized text fragments as shown in Figure 4.4(a). The HTML tagging of the CSIRO web-pages subdivides most documents into paragraphs containing only a small number of sentences. If the entity containment graph includes both document and paragraph vertices combined with their respective initial scores, the model allows to distinguish mentions of entities in unrelevant

parts of the document from those in more relevant paragraphs. The graph generation process was extended slightly for this experiment. After the initial document ranking, we selected all contained paragraph nodes that mention at least one candidate entity and ranked them as well according to the query terms. The graph was then built with both document and paragraph vertices. The last row of Table 4.2 shows that this graph modeling option indeed results in considerably better performance. All three propagation methods profit from the included paragraph vertices.

4.6 Experimental Study II: Entity Ranking on Wikipedia

The INEX entity ranking track meets our evaluation requirements with a testset of entity ranking topics and corresponding judgments on the INEX Wikipedia corpus. All topics specify a target entity type and a topic of interest in a few query terms. The target type is given as a Wikipedia category, e.g. "movies", "trees", or "programming languages". In contrast to other entity ranking tasks, each retrieved entity in the INEX track needs to have its own article in the Wikipedia collection. Obviously, this decision is only suitable for entity ranking within an encyclopedia, where we can assume that most mentioned entities in fact have their own entry. In consequence, a simple baseline run is given by a straightforward article ranking using the query terms that describe the topic of interest. Combined with an appropriate category filtering mechanism that also allows articles of descendant categories, such a baseline can reach already a high retrieval quality.

However, the described baseline approach shows no techniques so far that are specific to entity ranking. We want to evaluate in the following how the relevance propagation approach can be introduced to the setting of the Wikipedia entity ranking task. Furthermore, we extend the existing indegree propagation model by incorporating text fragments of various sizes.

4.6.1 Exploiting Document Entity Relations in Wikipedia

Entity mentions in Wikipedia articles are often linked towards their own encyclopedia entry. If we use these links to build a query-dependent entity containment graph, consisting of the top k initially retrieved entries and all their included linked entities, we can apply the introduced graph-based propagation models. Notice, that the graph does not distinguish between entity and document vertices in this case. Each vertex e represents an entity

and at the same time its text description in the corresponding Wikipedia entry. Hence, we also have an initial relevance estimation for each entity given by the score of its entry $w(e|q)$.

Initial experiments showed that the basic weighted indegree model does not improve over our initial baseline ranking. It even decreases the retrieval quality considerably. In fact, the direct text description of an entity is so important for the ranking that it needs to be considered in the retrieval model. Hence, we suggest the following extension of the weighted indegree:

$$PwIDG(e) = \lambda w(e|q) + (1-\lambda) \sum_{e' \in \Gamma(e)} w(e'|q).$$

The factor λ interpolates the initial article relevance with the summed relevance of other articles mentioning entity e. Since the above equation resembles the definition of a *personalized* graph centrality by incorporating the weighting of the vertices themselves, we call it personalized weighted indegree $PwIDG$.

Adding Smaller Sized Text Fragments Since the Wikipedia collection contains structured text, we can make use of the given paragraph segmentation and retrieve and score XML <P> elements as well. We have shown before, that the graph model allows to combine paragraph and article level relevance by simply adding vertices of both types to the graph (see Figure 4.4(a)). An entity e is then linked by other entities e' or paragraph vertices p when their text refers to e. For distinction, we denote the set of neighboring paragraph vertices of an entity e by $\Gamma^P(e)$, respectively $\Gamma^E(e)$ for the set of adjacent entities. In order to control the influence of both types of text fragments, a second interpolation factor μ is introduced:

$$PwIDG^*(e) = \lambda w(e|q) + (1-\lambda) \left(\mu \sum_{p \in \Gamma^P(e)} w(p|q) + (1-\mu) \sum_{e' \in \Gamma^E(e)} w(e'|q) \right).$$

The earlier introduced infinite random walk model (Section 4.4.2) can be extended equivalently with a second interpolation to control the propagation from articles or paragraphs.

Category Filtering For the categorization of entities, the INEX testset provides three files containing category name and identifier, the hierarchy of categories, and a list assigning each article to one ore more categories. The processing model for the ranking the entities of a given topic (see Section 4.2)

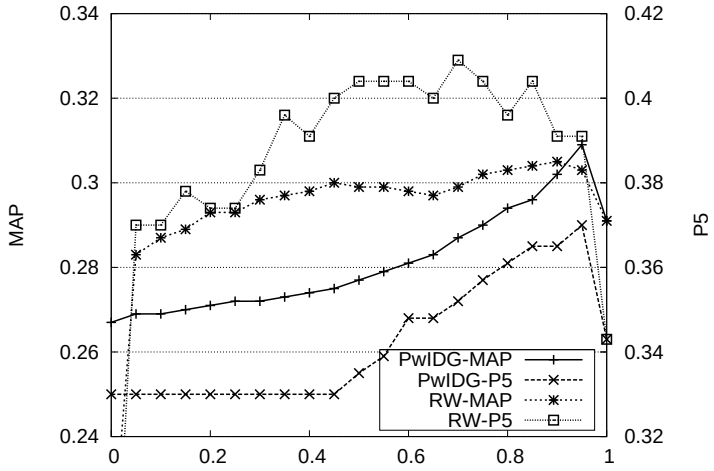


Figure 4.11: Influence of λ, $\mu = 0$

has to be extended by an additional filtering step to select only those entities belonging to the required target category. We found in the data that a Wikipedia article being assigned to the category "Italian composer" is not neccessarily also assigned to the parent categories "composer" or "Italian". When filtering entities, it is thus important to consider also descendant categories of the given target category. A training run on additional topics borrowed from the ad-hoc track showed that it was useful to include 3 generations of children, which became then the default setting for all reported experiments in this section.

4.6.2 Result Discussion

We generated for each INEX topic an entity containment graph from the top 200 articles retrieved by the title keywords. A standard language modeling retrieval model with Jelinek-Mercer smoothing was employed for the initial scoring of all text fragments. In difference to the expert finding task, no tagging at a preprocessing stage was needed, since the internal links in the corpus already mark mentions of entities within articles.

Setting of the Interpolation Factors Analyzing the introduced entity ranking models requires to study the influence of the two interpolation factors λ and μ. In a first test we only retrieved articles and were interested in finding an appropriate setting of λ for calculating a personalized weighted indegree $PwIDG$ as well as for the earlier introduced random walk model.

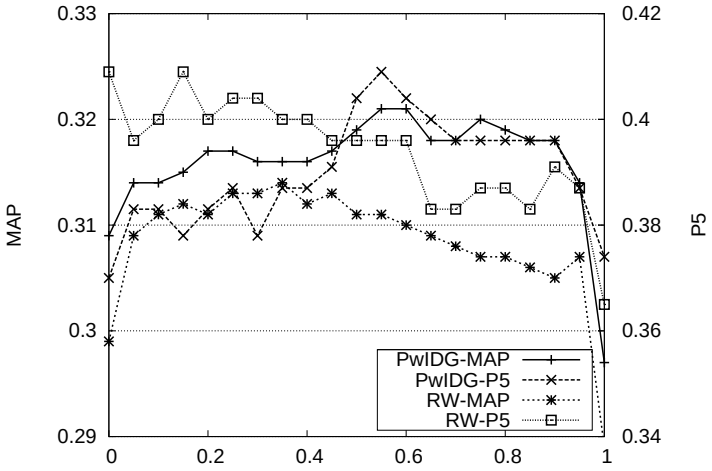


Figure 4.12: Influence of μ, λ set to best value

Figure 4.11 shows that λ needs to be set close to 1 for the indegree model, and to 0.7 to achieve the best random walk performance. Both results clearly point out the importance of the entity's own Wikipedia entry. On the other hand, the combination with the scores of other articles mentioning the entity clearly improves the retrieval quality.

For the second test, we kept λ fixed at its respective best values ($\lambda = 0.95$ for $PwIDG$, $\lambda = 0.7$ for RW), now varying the setting of μ, displayed in Figure 4.12. We can observe for the indegree model that mean average precision as well as precision on top of the retrieved list P@5 show a maximum when article and paragraph scores are considered equally with a setting of μ around 0.55. The random walk profits apparently less from the inclusion of further paragraph vertices. The retrieval quality drops soon when assigning a higher probability μ for propagation from paragraph vertices. The independence of λ and μ assumed by the testing procedure might be not adequate, but even without finding a global maximum the results indicate the advantage of the combining article and paragraph level relevance.

Comparison of Propagation Methods Table 4.3 shows the best outcome for all introduced ranking methods. It is important to see that our baseline, a simple ranking of all Wikipedia articles, achieves already a high mean average precision, which makes the task largely different from the before studied expert finding, where a direct entity ranking was impossible. It is thus not surprising that the baseline methods outperforms the weighted indegree, where the direct article ranking is not incorporated. By combining

	MAP	P@5
baseline	0.291	0.343
$wIDG$	0.267	0.330
$PwIDG$ $\lambda = 0.95$	0.309	0.370
$PwIDG^*$ $\lambda = 0.95, \mu = 0.55$	0.321	0.409
RW $\lambda = 0.7, \mu = 0.35$	0.314	0.400

Table 4.3: Overview: best setting per method

$PwIDG^*$	100	200	500
MAP	0.304	0.321	0.321
P@5	0.400	0.409	0.413

Table 4.4: Number of included top retrieved documents k

the baseline article score of an entity with those of text fragments mentioning the entity gives room for further improvements, but requires to set the interpolation controlled by λ and μ appropriately. The values presented in the table indicate the possible range of improvements that can be achieved by the different methods. It is not realistic to expect the exactly same improvements with the same parameter settings on a different testset and collection. When all informations sources are included, the best random walk model still stays behind the weighted indegree. We have to conclude therefore, that multi-step relevance propagation is not able to improve over the one-step model in case of the INEX task.

Graph Building and Category Filtering Controlling the number of top retrieved documents that are considered when building the entity containment graphs was an important parameter in the previous study on expert finding. The INEX track shows in this respect similar characteristics as the expert finding task on the CSIRO corpus. Altough the number of relevant entities per topic is considerably higher for the INEX task, the achieved recall is not growing noticably if we include more than 200 articles in the graph building as shown in Table 4.4.

Table 4.5 confirms the need to extend the list of given target categories

$PwIDG^*$	0	1	2	3	4	5
MAP	0.272	0.298	0.331	0.321	0.316	0.303
P@5	0.322	0.387	0.430	0.409	0.391	0.374

Table 4.5: Number of included child categories

by their child categories. Including up to 2 generations of child categories yields considerable improvements on mean average precision as well as for precision at 5 retrieved entities. Including further descendants has a slightly negative effect on precision. Remember that the same test on the training set suggested to include 3 generations of child categories. In consequence, all results reported in this section could be improved equally by employing a more selective category filtering.

4.7 Searching Mixed-typed Entities

In this last section, we discuss and evaluate ideas for creating a mixed-typed topic overview. Thus, we address here entity ranking task 1 (see Section 4.1). Compared to other entity ranking tasks studied in the previous sections, this one represents the most open information need. We think mainly of a user who wants to get a first overview on a topic he/she is not familiar with. In response to an initial ad-hoc query, we do not return documents, but entities of different types that are strongly related to the topic. Such entity overview provides a fast access to capture the essence of a topic. The entities might be used further in subsequent new queries, or for new kinds of browsing interface as known from faceted search (Yee et al., 2003; Bast et al., 2007). Imagine that a user is looking for the "Life of Pablo Picasso" or "Egyptian Pyramids", the returned topic overview should contain associated `people`, `countries` or `dates`.

In particular we demonstrate how to deal with a large and heterogeneous set of types, some being more generic, others rather specific. Such type set is typical when working with NLP techniques such as named entity recognition or semantic tagging. Since the user query does not specify a certain type, the system has to decide itself whether specific types should be preferred over others.

4.7.1 Model Adaptations

The task of searching mixed entity types requires the following adaptations of the previously introduced graph models and ranking schemes.

Entity Containment Graphs for Mixed-Typed Entities Since we do not want to exclude any types in advance, our entity containment graph has to contain all found entities of all types. However, this would result in almost unhandable graph sizes, when using roughly the same number of relevant documents for the graph generation as in previous experiments. Moreover,

we found a high number of topically irrelevant entities in such large graphs. In order to tackle the problem, we could clearly reduce the number of top relevant documents in the graph building process. However, such a solution would exclude entities not mentioned in the first few documents and the graph would still contain many irrelevant entities if those documents are not exclusively about the specified topic. We choose therefore to switch from document to sentence retrieval for this task. In contrast to entire documents, the entities of a relevant sentence have a clearer connection to the given topic. Furthermore, sentence retrieval provides a better control of the graph sizes. Documents vary highly in length and hence also in the number of mentioned entities. By specifying the number of top retrieved sentences instead, we get a better dimensioning of the total number of entities in the graph model and achieve a higher topicality of the included entities.

Adapting the Indegree Model Once the graph is constructed, the proposed relevance propagation techniques can be used. We observed, however, in first experiments that all degree-dependent methods are biased by a few entities of very general types, such as descriptions or country names. Those entities are usually not specific to the topic but have a high frequency of mentions.

An ad-hoc method to overcome the problem consists in removing all generic types from the final result list. Notice that we do not want to exclude them already in the graph building phase since their connectivity supports the relevance propagation in the graph. The ad-hoc solution is, however, only applicable if we know a priori which types can be regarded as too generic and non informative.

We suggest therefore also a second method, which is inspired by the inverse document frequency component in document retrieval models. The inverse document frequency is used to re-weight terms with respect to their specifity. Often occurring terms with stop-word characteristics are effectively disregarded. Similarly, we calculate the inverse sentence frequency $isf(e)$ of an entity and combine it with the result of the graph-based entity score, for instance with its indegree $IDG(e)$:

$$\begin{aligned} isf(e) &= \log\left(\frac{|S|}{|\{s \in S | s \text{ contains } e\}|}\right), \\ RSV(e) &= IDG(e)\, isf(e), \end{aligned}$$

with S being the set of all sentences s in the collection. The final entity score $RSV(e)$ reflects now also the specifity of an entity with respect to the given query topic. Notice that the sentence frequency of an entity is equivalent to the indegree of the entity in the global, query-independent entity

containment graph. The suggested method therefore stresses the differences between global and local graph.

4.7.2 Experiments

For testing the presented ideas, we could not use a standard query and judgment set from the evaluation initiatives, since they do not consider the task of open-domain entity ranking. Instead we built up our own evaluation environment and asked a number of test users to formulate queries and to judge the relevance of the returned entities. The results of this study have to be regarded preliminary since we could not judge a large pool of different rankings and queries.

Collection and Tagging Again the Wikipedia collection was used, since it represents an interesting source for open-domain entity ranking queries. For tagging, the open source SuperSense Tagger[3] was trained on the *BBN Pronoun Coreference and Entity Type Corpus*, which includes the annotation of nominal types (like `Person`, `Facility`, `Organization`), and numeric types (like `Date`, `Time`, `Percent`). Further description types are dedicated to identify common nouns that refer or describe named entities. For example, the words "father" and "artist" would be tagged as a `person-description`.

Tagging the entire Wikipedia corpus resulted in 28 million occurrences of 5,5 million unique entities. The tagged corpus is publicly available attached with a full reference of all tagged types (Zaragoza et al., 2007a).

Evaluation The evaluation environment for the user study was set up in the following way. The user first had to choose a topic and to formulate an initial term query. There was no restriction on the choice of the topic other than then to remind the user that his/her query would be run on a collection of encyclopedia texts, such as the Wikipedia corpus. The user was also reminded that he/she should feel knowledgeable on the chosen topic to be able to later judge the relevance of entities.

The system employed the Lucene search engine[4] to retrieve the 500 most relevant sentences from the collection. It should be mentioned that the applied scoring function does not provide probabilistic scores here. However, we still ensured a probabilistic score range by normalising the scores of the top retrieved included sentences. All mentioned entities were ranked according to their indegree $IDG(e)$ and presented to the user for evaluation.

[3] `http://sourceforge.net/projects/supersensetag/`

[4] see `http://lucene.apache.org/`

Query	**"Yahoo! Search Engine"**
Most Important	Yahoo, Google, MSN, Inktomi, Yahoo.com
Important	Web, crawler, 2004, AltaVista, 2002, Amazon.com, Jeeves, TrustRank, WebCrawler, Search Engine Placement, more than 20 billion Web, eBay, Worl Wide Web, BT OpenWorld, between 1997 and 1999, Stanford University and Yahoo, AOL, Kelkoo, Konfabulator, AlltheWeb, Excite
Related	users, Firefox, Teoma, LookSmart, Widget, companies, company, Dogpile, user, Searchen Networks, MetaCrawler, Fitzmas, Hotbot, ...
Query	**"Budapest"**
Most Important	Budapest, Hungary, Hungarian, city, Greater Budapest, capital, Danube, Budapesti Kzgazdasgtudomnyi s llamigazgatsi Egyetem, M3 Line, Pest county
Important	University of Budapest, Austria, town, Budapest Metro, Soviet, 1956, Ferenc Joachim, Karl Marx University of Economic Sciences, Budapest University of Economic Sciences, Etvs Lornd University of Budapest, Technical University of Budapest, 1895, February 13, Budapest Stock Exchange, Kispest, ...
Related	Paris, Vienna, German, Prague, London, Munich, Collegium Budapest, government, Jewish, Nazi, 1950, Debrecen, 1977, M3, center, Tokyo, World War II, New York, Zagreb, Leipzig, population, residences, state, cementery, Serbian, Novi Sad, 1949, Szeged, Turin, Graz, 6-3, Medgyessy, ...
Query	**"Tutankhamun curse"**
Most Important	Tutankhamun, Carnarvon, mummies, Boy Pharaoh, The Curse, archaeologist, Howard Carter, 1922
Important	Pharaohs, King Tutankhamun
Related	Valley, KV62, Curse of Tutankhamun, Curse, King, Mummy's Curse, ...

Table 4.6: Example queries and user judgements of the entities

For judging the retrieved entities, the user was shown the entire returned list of entities in ranked order. Each of the entities should then be assigned one of the following labels: *Most Important*, *Important*, *Related*, *Unrelated*, or *Don't know*. The user was asked to assess all entries if possible, and at least the first fifty. The users were not given any further instructions nor were they trained before using the evaluation system. 10 test persons were recruited and each judged from 3 to 10 queries, coming to a total of 50 judged queries.

Some of the gathered queries and judgments are shown in Table 4.6. With those examples we want to demonstrate the difficulty and subjectivity of the evaluation. The machine tagging and entity ranking indeed delivered interesting and highly related entities, but the quality is not always as expected. The ranked list often shows up different names and spellings of the same entity like "Yahoo" and "Yahoo.com", or "Tutankhamun" and "King Tu-

MODEL	MAP	P@10	DCG	nDCG
wMAX	0.34	0.37	67.91	0.64
IDG	0.50	0.54	79.69	0.78
wIDG	0.48	0.51	79.11	0.76
IDG filt.	0.50	0.52	79.23	0.79
IDG isf	**0.60**	**0.63**	**83.89**	**0.84**
wIDG isf	0.54	**0.63**	82.68	0.81

Table 4.7: Performance of the different models

tankhamun". It also contains many items that are hard to judge without the context of the surrounding sentence as dates or numbers. We were aware that all those problems influence the quality of the judgments, but still regard them as sufficient for a preliminary testing of the above shown approaches.

Apart from mean average precision (MAP) and the precision at 10 retrieved entities (P@10), we wanted to use a measure for graded relevance judgments, since the entity ranking task clearly asks for a finer-grained distinction of relevance. Therefore the most established graded relevance measure, the (normalized) discounted cumulative gain (nDCG) suggested by Järvelin and Kekäläinen (2002), was used with the gain vector $\{10, 3, 1, 0\}$ corresponding to the 4 judgment labels. All entities labeled as *Don't know* were ignored in the ranking. For the binary relevance measures, only *Important* and *Most Important* marked entities were considered as relevant, whereas all others entities were treated as irrelevant.

Results Table 4.7 gives an overview of the results. The best results are shown in bold face. Looking first on the graph based retrieval models from previous sections, the indegree again outperforms the simplistic *wMAX*. However, in contrast to previous experiments, the weighting by initial sentence scores (*wIDG*) slightly decreases the retrieval quality of the indegree model. We tried several score normalization techniques, but were unable to achieve the expected increase from score propagation. The reasons for this difference are difficult to explain. Firstly, we employed a different retrieval system for the experiments in this section, which does not return probabilistic scores. Secondly, the scores of the top retrieved sentences vary more than those of the top retrieved documents examined before. Multi-step relevance propagation, consequently, did not work in this scenario either. Therefore, we did not even list the results for random walk based models in the table.

On the other hand, our proposed *isf* component clearly shows improvements on all measurements. It also achieves better results than the ad-hoc filtering of generic types (*IDG* filt.). The improvements in our preliminary

study are clear enough, to expect the *isf* component to be an appropriate extension also for other graph based entity ranking models.

4.8 Summary and Conclusions

We motivated and defined the retrieval of entities which differs in some important ways from the retrieval of any other kind of text fragments. The most obvious difference concerns the distinction of an entity's identity and its mentions in the text. Moreover, entities cannot be ranked directly based on their text representation. We distinguished in the following three subtasks of entity ranking, differing in whether the topic and entity type are specified or left open, namely open domain entity search, typed search, and list completion.

Our own approach to entity ranking is based on graphs. We showed how the relation between documents and entities can be modeled in bipartite entity containment graphs. The graph modeling is flexible enough to capture also other relevance related information, like the strength of association between entities and documents, or the relation of entities among themselves if it is known from a given ontology.

Formulating the task of entity ranking as a graph-based relevance propagation has shown to be a fruitful theoretical model. It does not only motivate and justify existing propagation models used for expert finding or question answering, but also suggests to extend those simple models by exploiting more graph features. We showed how the basic indegree model can be extended to a random walk, which also takes into account the relevance of indirect neighbors.

Both graph modeling and relevance propagation has been tested on different collections and entity ranking tasks. The main findings of the experimental studies can be summarized as follows. The pure unweighted graph structure of the entity containment graph provides useful additional hints for the ranking of entities, but cannot come up with high quality rankings on its own. Similarly, our baseline ranking approach that relies solely on the initial ranking of documents is outperformed by all graph-based relevance propagation models. It is thus necessary to combine both the structural features of the graph as well as the initial document ranking to yield the best retrieval performance. From the tested relevance propagation models the weighted indegree model have shown the most robust performance. Without the tuning of further parameters it came in all cases close to the best performing model. The probabilistic random walk suffers slightly from its normalized edge transition weights, which do not model the propagation appropriately.

The HITS-like normalization is able to solve the problem and shows that random walks are able to achieve slightly superior rankings than the simple indegree, but come with the disadvantage of a more costly computational model and further parameters that require to be set appropriately.

By experimenting with different collections and tasks like expert finding or the search for Wikipedia entities we demonstrated the usefulness of a uniform model for entity ranking. Instead of developing a ranking model for each single task our graph-based entity ranking framework needed only slight adaptations to work in different environments.

We could not test all suggested options for graph modeling with the given collections and tasks. Especially the incorporation of known relations between the entities themselves seems an interesting direction for future work. If ontologies are available for entities in Wikipedia, or organizational structures of the enterprise are known in case of expert finding, the modeling of this additional information might improve the retrieval results further.

Another interesting direction for future work lies in the challenging task of finding suitable short text fragments supporting the estimated ranking of entities. Although we explained in the introduction that the added value of entity ranking compared to passage or XML retrieval is that it directly returns the extracted ranked list of entities, it is important to notice that such a result list is in many cases only useful in combination with links to supporting sentences or passages. Our graph-based propagation models might be useful here as well, since they also rank the text fragments with respect to the included entities.

5

Review and Outlook

Whereas it is common practice to repeat the main achievements and to point out their contribution to the research community at the end of any scientific work, we think we satisfied this issue already with the summaries and conclusions at the end of each chapter. The interested reader is hence referred to the respective last sections of each chapter. Instead, this last chapter gives a more critical review on the presented work and concludes by an outlook on possibilities to integrate and combine the research shown in the three main chapters of this thesis.

5.1 A critical Review

At this point we take a critical perspective when judging the presented work again on the basis of the research objectives proposed in the introduction. The following review is meant to point at the weak points and limitations of the proposed work, which we think should be a requirement for all scientific publications. At the same time, the critical review shows perspectives for future work.

Context Refined Document Retrieval Our first goal in Chapter 2 is to refine retrieval by taking into account the context of a user. However, the context of a user is a rather comprehensive and vague concept, which needs to become more precise. Hence, we try here to take certain aspects of context information into account which we expect to have a concrete influence on relevance. If we take a look again at the considered context dimensions, like topicality, genre, date, or location (see Section 2.1), the question arises whether they really describe the "user context" and not something that would better be called "search context". In order to demonstrate the difference, one

can think of a user sitting at home and planning the next vacation. A web search for accommodations should not consider the user context sitting at home, but the context of the search planning a vacation at a certain location. The difference between user context and search context is mentioned but not investigated, yet. It might even be appropriate to substitute the user by the search context, for the purpose of improving retrieval results.

We address Objective (A1) by introducing conceptual language models as a generic framework for the modeling of context information (see Section 2.1.1). It is argued why a set of concept models describes the search context better than individual user models. We also motivated the idea of using language models as a representation of contextual concepts. The advantage of the generic framework based on language models becomes apparent when scoring documents against the given context information. The language models are representation and classifier at the same time. Also the score combination problem is simplified when all involved scores result from the same retrieval model. When applying a retrieval model that normalizes by the query length, like the NLLR, we even achieve individual classification scores lying in the same value range (see Section 2.2). Unfortunately, the testing remains rather limited here. The choice of the considered context dimensions is less driven by the characteristics of the users' context, but by the available classifiers and meta-data coming together with the evaluation corpus. In fact, the use of conceptual language models is shown only on two example dimensions: topicality and location (see Section 2.3). Moreover, the location dimension could not be modeled appropriately due to the broad location categories considered by the evaluation track. It is thus still necessary to investigate experimentally how suitable language models represent and categorize locations, genres or readability. The tests on the selected context dimensions indicated that some dimensions have a higher influence on the retrieval quality than others. Hence, they are more useful when specifying the user's context. However, this does not allow to draw any conclusions on the usefulness of a certain context dimension on the level of individual queries.

In contrast to the score combination of all context dimensions, the combination with the relevance of the initial term query as described in Objective (A2) is not sufficiently solved, yet. The experiments use different normalization techniques when working with different collections (see Section 2.2.1 and Section 2.5.4), in order to adjust to the respective setting. More testing that also considers other combination models is necessary here. It is further important to find appropriate ways of taking "dislike" statements into account, since they play an important role in explicit user feedback.

The new method of relevance feedback suggested in response to Objec-

tive (A3) overcomes several problems associated with common feedback on documents or suggested query terms (see Section 2.4). Document feedback is time consuming and suggesting additional query terms requires more knowledge about the search topic than the query profile based feedback introduced here. Query profiles further enable to distinguish on the base of an individual query which context dimensions have a meaningful impact for query clarification, and are therefore suited for refinement. Hence, the suggested query profile based feedback can shorten the user interaction by asking only necessary clarification questions (see Section 2.5). In order to claim that the proposed relevance feedback is more helpful from a user point of view, it is not sufficient to show that the method improves the retrieval quality on a given test collection. The retrieval improvements need to be compared to those of other feedback methods. Furthermore, it would be interesting to see user studies that try to measure the more subjective satisfaction of use in comparison to a system without feedback or systems using other forms of feedback.

Structured Retrieval on XML If retrieval should take structural constraints into account, it is important to ask what kind of structural features are meaningful from a retrieval perspective. The discussion at the beginning of Chapter 3 starts from a slightly different point of view. Instead of examining the impact of structural features on retrieval, the reasoning starts from a data-centric perspective. We look here in fact more on the properties of XML data with the question how the given structural mark-up can be used for query refinement, than from the opposite side with the question what structural features help retrieval. The same criticism holds for the existing query languages NEXI and XQuery Full-text. The proposed integration of the two languages does not overcome the data-bound perspective either (see Section 3.1.5). When the NEXI embedding in XQuery is suggested, the proposal is developed with mainly practical issues in mind. The more restricted search features of NEXI are easier met by sound retrieval models, and the composition with other XQuery expressions still provides a rather powerful query language.

All research objectives in the area of structured retrieval address performance issues. The presented index structure for the support of XML retrieval concerned with Objective (B1) carefully avoids "data-independent" redundancy (see Section 3.2.3). It also enables fast positional access to the data, which minimizes the unavoidable random access costs. However, conventional inverted indices make use of further compression techniques that allow to reduce redundancy in the data as well. Most important, the number

of bits that encode the position of a term can be reduced. The compression ratio achieved by such techniques on conventional indices is tremendous. If we aim at making XML retrieval feasible on the same collection sizes, it is necessary to incorporate such compression techniques also into the here presented index structure.

Our optimized containment join algorithm performs clearly better than other structural joins in common retrieval situations (see Section 3.3.1). It is admissible here to focus on the often occurring cases, and to require a longer evaluation for special cases. We showed that the containment evaluation can indeed take place at query time and does not need to be pre-computed in a large index. The execution time of the optimized structural join remains in the same time range as the score calculation and aggregation. Especially the later score aggregation considerably exceeds the execution time of the containment join in some cases. Hence, optimizing the aggregation as well would be the next step to satisfy Objective (B2).

Query optimization addressed by Objective (B3) is analyzed on the physical level for simple but often occurring query patterns and on logical level for complex queries. On the logical level, however, the effect of optimization is only demonstrated on the base of a single query pattern (see Section 3.4). Though the pattern is probably typical for structured queries, it is unknown whether the outcome is representative for the effect of logical query optimization in structured query languages. In order to introduce a generic approach for logical query optimization the proposed cost modeling needs to be extended considerably. Still, the shown example motivates future research in this direction.

For the physical query execution of simple query patterns, three alternative query plans are considered that do not differ in the order of base operations, but follow either a joint or a split processing model (see Section 3.3.3). It is argued why changing the order of the involved operations is not expected to improve efficiency in this case. In difference to the logical level, the analysis is driven by the assumption to find a single best query plan in all situations, and not a situation dependent optimization strategy. The only considered parameter for a possible optimization strategy is the number of query terms, which has a direct influence on the number of operations in the split processing model. The analysis shows here that the operation number plays in fact a minor role. However, the shortly addressed score aggregation strategy becomes crucial when evaluating longer queries.

Entity Ranking The aim to develop a generic framework, that allows to rank entities by their probability of relevance, is a rather wide-ranging task

that asks to solve several sub-problems. In this case, a graph-based relevance propagation approach is chosen, which includes finding a suitable a graph model and relevance propagation.

The proposed graph modeling (see Section 4.3) represents the relation of text fragments and entities in a bipartite entity containment graph as requested by Objective (C1). It also considers a number of options to incorporate additional information, like association weights, relevance scores on different levels of text granularity, or ontologies showing the relations of entities among themselves. The graph modeling provides thus a high flexibility to model the different cases of entity ranking. Unfortunately, it was not possible to test all suggested modeling options on the given test collections. Especially the integration of ontologies seems an interesting direction for further research.

We addressed Objective (C2) by suggesting a number of graph-based relevance propagation models, some based on existing non-graph-based approaches, others transferred from web retrieval on link graphs (see Section 4.4). One of the main advantages of the graph-based ranking approach is the ability to rank an entity node not only by text fragments mentioning it, but also by its indirect neighbors in the graph. The introduced random walks model this relevance propagation from indirect neighbors (see Section 4.4.2). Interestingly, the results on the expert finding task show only a slightly better ranking compared to the simple propagation models that take only the relevance of direct neighbors into account (see Section 4.5.1). Hence, the experimental results reported here make it questionable whether a graph-based ranking approach is really necessary for all entity retrieval tasks. Other tasks might still show more need for graph-based propagation. For timeline retrieval for instance, the ranking of date entities is probably helped by looking at the co-occurrence with other more meaningful person or location entities. In that case, graph-based ranking would play a more important role again.

All propagation models take the association weights on the document-entity edges into account. It seems logical to consider different association weights depending on the relation between document and entity. Experimenting with those weights, however, yielded no measurable improvements (see Section 4.5.1). Even in the case of structured email data, where it seems reasonable to expect a closer relation between the email author and the text than between the text and any other mentioned persons, we could neither find a better weight setting ourselves, nor could we confirm the positive effects of weight settings reported by others working on the same collection. It seems, however, unlikely that a uniform setting of association weights is indeed the best possible setting here. Thus we regard it necessary to repeat

the test on another collection, maybe employing machine learning techniques to find the best weight setting.

Since some propagation approaches are probabilistic in nature, it is sound to also apply a probabilistic scoring function for estimating the initial relevance of the text fragments (see Section 4.4). On the other hand, it would have been interesting to test the actual influence of the initial scoring on the relevance propagation, and to consider alternative scoring models here as well.

5.2 Outlook on Possibilities of Combination and Integration

The contribution of this thesis is clearly divided over the three main chapters. This final section gives an integrated view by asking how the different proposed methods and techniques can learn from each other.

Can Entity Retrieval Learn from Structure and Context? Structural features are used already in entity retrieval. In the test on association weighting, the weights were determined with respect to the text element the entity is mentioned in (see Section 4.5.1). Since the email collection came with a meaningful structure, e.g. marking authors and recipients of emails, we tried to make use of that structural information for the weight setting. Structural features have been exploited also in a second more successful way. We considered text fragments of different granularity when building entity containment graphs and combined this way the initial relevance of paragraphs and entire documents (see Section 4.6.2). The more focussed paragraph relevance improved the precision of our entity ranking, the document relevance increased the recall.

Entity retrieval still needs an appropriate query language and efficient execution. One imaginable approach towards an effective expert level query language would be to introduce a set of new XQuery functions similar to the ones used for the full-text search features, but ranking entities instead of elements. The integration of entity search functionality in XML retrieval systems might also help to improve the efficiency of entity retrieval. The necessary efficient support for score propagation and combination is already available in structured retrieval systems.

Furthermore, we see two possibilities for context aware query refinement in the entity retrieval process: (1) Instead of applying a standard document ranking to find the most relevant text fragments, the initial text relevance

estimation can incorporate more context information if available. (2) We can further introduce a feedback step in the entity retrieval process. The graph-based propagation framework allows to represent relevance by its vertex weighting. Hence, feedback information can be integrated naturally.

Can Structured Retrieval Become Context Refined? We stated already before that structured retrieval needs to be brought back from its current data-centric point of view towards more usage oriented applications. An obvious direction for future research would then be to analyze which structural features play a role in which search context.

Thinking of the query profiles based feedback suggested in Chapter 2, such technique might be useful for structural search constraints as well. A structural query profile would be able to display to the user the most influential element names or rooted paths to relevant elements for a given query. It is then necessary to develop an interface that allows to refine the initial query by user selected structural constraints seen in the profile. Such technique also overcomes the problem that users often have little knowledge about the actual structure of documents in the collection and therefore cannot write structural queries, or choose unwanted inaccurate restrictions.

Can Entity Retrieval Improve Context Awareness? We just suggested how structural features can be integrated in a query profile based refinement strategy. Entities might be even more interesting for query refinement. Entity mentions are rather distinctive and meaningful units of text, and typically kept when reducing a text to a short summary. Therefore, entities are highly suitable for feedback and query refinement.

There are, in fact, existing approaches in this direction (Yee et al., 2003; Bast et al., 2007). User interfaces for so-called faceted search offer browsing facilities to explore the data. They display entities mentioned in related texts together with the number of occurrences, a kind of entity profile in our terminology. Such interfaces might be improved by applying the entity ranking methods proposed in this thesis, instead of relying simply on the number of occurrences. Our entity ranking approach would even be helpful in two ways. It improves the relevance estimation of entities, and provides furthermore a ranking of supporting text fragments, that can be displayed in response to the selection of an entity.

Bibliography

A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 14–23, New York, NY, USA, 2006. ACM Press.

S. Al-Khalifa, C. Yu, and H. V. Jagadish. Querying structured text in an XML database. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 4–15, New York, NY, USA, 2003. ACM Press.

J. Allan. HARD Track Overview in TREC 2003: High Accuracy Retrieval from Documents. In *Proceedings the 12th Text REtrieval Conference (TREC)*, pages 24–37, 2003.

J. Allan. HARD Track Overview in TREC 2004: High Accuracy Retrieval from Documents. In *Proceedings the 13th Text REtrieval Conference (TREC)*, pages 25–35, 2004.

S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, M. Holstege, J. Melton, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 Full-Text 1.0 Working Draft. W3C, published online at `http://www.w3.org/TR/2007/WD-xpath-full-text-10-20070518/`, 2007.

V. N. Anh and A. Moffat. Pruned query evaluation using pre-computed impacts. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 372–379, New York, NY, USA, 2006. ACM.

K. Balog and M. de Rijke. Finding Experts and their Details in E-mail

Corpora. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 1035–1036. ACM, 2006.

K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In Efthimiadis et al. (2006), pages 43–50.

H. Bast, A. Chitea, F. Suchanek, and I. Weber. ESTER: efficient search on text, entities, and relations. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 671–678, New York, NY, USA, 2007. ACM.

T. Bauer and D. B. Leake. Real Time User Context Modeling for Information Retrieval Agents. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, pages 568–570. ACM, New York, NY, USA, 2001.

N. J. Belkin. Interaction with Texts: Information Retrieval as Information-Seeking Behavior. In *Information Retrieval '93, Von der Modellierung zur Anwendung*, pages 55–66. Universitaetsverlag Konstanz, 1993.

N. J. Belkin, D. Kelly, H.-J. Lee, Y.-L. Li, G. Muresan, M.-C. Tang, X.-J. Yuan, and X.-M. Zhang. Rutgers' HARD and Web Interactive Track Experiences at TREC 2003. In *Proceedings the 12th Text REtrieval Conference (TREC)*, pages 418–429, Gaithersburg, MD, USA, 2003. NIST.

K. Bharat and M. R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In *SIGIR*, pages 104–111. ACM, 1998.

H. M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors. *Intelligent Search on XML Data, Applications, Languages, Models, Implementations, and Benchmarks*, volume 2818 of *Lecture Notes in Computer Science*, 2003. Springer.

P. Boncz. *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*. PhD thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, May 2002.

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 *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 479–490, New York, NY, USA, 2006. ACM.

P. A. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teubner. Pathfinder: XQuery - The Relational Way. In K. B"ohm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, editors, *VLDB*, pages 1322–1325. ACM, 2005.

U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*, 2005. Springer.

A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

F. J. Burkowski. Retrieval activities in a database consisting of heterogeneous collections of structured text. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 112–125, New York, NY, USA, 1992. ACM.

C. S. Campbell, P. P. Maglio, A. Cozzi, and B. Dom. Expertise identification using email communications. In *CIKM*, pages 528–531. ACM, 2003.

S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors, *WWW*, pages 571–580. ACM, 2007.

S. Chakrabarti, K. Puniyani, and S. Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 717–726. ACM, 2006.

C. Chatfiled. *The Analysis of Time Series*. Chapman and Hall, 3rd edition edition, 1984.

H. Chen, H. Shen, J. Xiong, S. Tan, and X. Cheng. Social Network Structure behind the Mailing Lists: ICT-IIIS at TREC 2006 Expert Finding Track. In *Proceedings the 15th Text REtrieval Conference (TREC)*, 2006.

Y. Chen and J. Martin. Towards Robust Unsupervised Personal Name Disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 190–198, 2007.

P.-A. Chirita, J. Diederich, and W. Nejdl. MailRank: using ranking for spam detection. In Herzog et al. (2005), pages 373–380.

C. L. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365, New York, NY, USA, 2001. ACM.

N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the TREC-2005 Enterprise Track. In *Proceedings the 14th Text REtrieval Conference (TREC)*, 2005.

B. W. Croft. Combining Approaches to Information Retrieval. In B. W. Croft, editor, *Advances in Information Retrieval : Recent Research From the Center for Intelligent Information Retrieval*, pages 1–36. Kluwer Academic Publishers, New York, 2002.

H. Cunningham. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.

H. T. Dang, J. Lin, and D. Kelly. Overview of the TREC 2006 Question Answering Track. In *Proceedings the 15th Text REtrieval Conference (TREC)*, 2006.

F. Diaz and J. Allan. Browsing-based User Language Models for Information Retrieval. Technical report, CIIR University of Massachusetts, 2003.

F. Diaz and R. Jones. Using Temporal Profiles of Queries for Precision Prediction. In M. Sanderson, K. Järvelin, J. Allan, and P. Bruza, editors, *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–24. ACM, Sheffield, UK, 2004.

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: is more always better? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, New York, NY, USA, 2002. ACM.

S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: a system for personal information retrieval and re-use. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 72–79, New York, NY, USA, 2003. ACM.

E. N. Efthimiadis, S. T. Dumais, D. Hawking, and K. Järvelin, editors. *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, 2006. ACM.

R. A. Elmasri, S. B. Navathe, and C. Shanklin. *Fundamentals of Database Systems*. Addison-Wesley Longman Publishing Co., Inc., 1999.

G. Erkan and D. R. Radev. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479, 2004.

M. Fernández, A. Malhotra, J. Marsh, M. Nagy, and N. Walsh. XQuery 1.0 and XPath 2.0 Data Model (XDM). W3C, published online at `http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/`, 2007.

N. Fuhr and K. Groβjohann. XIRQL: An XML query language based on information retrieval concepts. *ACM Trans. Inf. Syst.*, 22(2):313–356, 2004.

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, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*, volume 3493 of *Lecture Notes in Computer Science*, 2005. Springer.

R. Godin, J. Gecsei, and C. Pichet. Design of a Browsing Interface for Information Retrieval. In N. J. Belkin and C. van Rijsbergen, editors, *SIGIR'89, 12th International Conference on Research and Development in Information Retrieval, Cambridge, Massachusetts, USA, June 25-28, 1989, Proceedings*, pages 32–39. ACM, 1989.

R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *VLDB*, pages 436–445. Morgan Kaufmann, 1997.

T. Grust and M. van Keulen. Tree Awareness for Relational DBMS Kernels: Staircase Join. In Blanken et al. (2003), pages 231–245.

T. Grust, M. van Keulen, and J. Teubner. Accelerating XPath evaluation in any RDBMS. *ACM Trans. Database Syst.*, 29:91–131, 2004.

R. Gunning. The Fog Index After Twenty Years. *Journal of Business Communication*, 6(2):3–13, 1968.

D. Harman. Relevance Feedback Revisited. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pages 1–10. ACM, 1992.

D. He and D. Demner-Fushman. HARD Experiment at Maryland: from Need Negotiation to Automated HARD Process. In *Proceedings the 12th Text REtrieval Conference (TREC)*, pages 707–714, Gaithersburg, MD, USA, 2003. NIST.

M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring Index Quality Using Random Walks on the Web. *Computer Networks*, 31 (11-16):1291–1303, 1999.

O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, and W. Teiken, editors. *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, 2005. ACM.

D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: ad-hoc and cross-language track. In *Proceedings the 7th Text REtrieval Conference (TREC)*, pages 227–238. NIST, 1998.

D. Hiemstra, H. Rode, R. van Os, and J. Flokstra. PFTijah: text search in an XML database system. In *Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR), Seattle, WA, USA*, pages 12–17. Ecole Nationale Supérieure des Mines de Saint-Etienne, 2006.

E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. Question Answering in Webclopedia. In *Proceedings the 9th Text REtrieval Conference (TREC)*, 2000.

G. Hu, J. Liu, H. Li, Y. Cao, J.-Y. Nie, and J. Gao. A Supervised Learning Approach to Entity Search. In *Information Retrieval Technology*, Lecture Notes in Computer Science, pages 54–66, 2006.

N. A. Jaleel, A. Corrada-Emmanuel, Q. Li, X. Liu, C. Wade, and J. Allan. UMass at Trec2003: HARD and QA. In *Proceedings the 12th Text REtrieval Conference (TREC)*, pages 715–725, Gaithersburg, MD, USA, 2003. NIST.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Articulating information needs in XML query languages. *ACM Trans. Inf. Syst.*, 24(4): 407–436, 2006.

R. Kaushik, R. Krishnamurthy, J. F. Naughton, and R. Ramakrishnan. On the Integration of Structure Indexes and Inverted Lists. In G. Weikum, A. C. König, and S. Deßloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 779–790. ACM, 2004.

J. Kazama and K. Torisawa. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 698–707, 2007.

J. M. Kleinberg. Bursty and Hierarchical Structure in Streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *SODA*, pages 668–677, 1998.

J. Ko, E. Nyberg, and L. Si. A probabilistic graphical model for joint answer ranking in question answering. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 343–350, New York, NY, USA, 2007. ACM.

D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. Centrality Indices. In Brandes and Erlebach (2005), pages 16–61.

D. Koschützki, K. A. Lehmann, D. Tenfelde-Podehl, and O. Zlotowski. Advanced Centrality Concepts. In Brandes and Erlebach (2005), pages 83–111.

W. Kraaij. *Variations on language modeling for information retrieval*. PhD thesis, University of Twente, Netherlands, 2004.

A. Kritikopoulos, M. Sideri, and I. Varlamis. BlogRank: ranking weblogs based on connectivity and similarity features. In *AAA-IDEA '06: Proceedings of the 2nd international workshop on Advanced architectures and algorithms for internet delivery and applications*, page 8, New York, NY, USA, 2006. ACM Press.

O. Kurland and L. Lee. PageRank without hyperlinks: structural re-ranking using links induced by language models. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*, pages 306–313. ACM, 2005.

O. Kurland and L. Lee. Respect my authority!: HITS without hyperlinks, utilizing cluster-based language models. In Efthimiadis et al. (2006), pages 83–90.

B. Larsen, A. Tombros, and S. Malik. Is XML retrieval meaningful to users?: searcher preferences for full documents vs. elements. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 663–664, New York, NY, USA, 2006. ACM.

Q. Li and B. Moon. Indexing and Querying XML Data for Regular Path Expressions. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *VLDB*, pages 361–370. Morgan Kaufmann, 2001.

J. Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, 25(2):6, 2007.

J. Lin and B. Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 116–123, New York, NY, USA, 2003. ACM Press.

X. Liu, B. W. Croft, and M. B. Koll. Finding experts in community-based question-answering services. In Herzog et al. (2005), pages 315–316.

C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In Yu et al. (2006), pages 387–396.

S. Malik, C.-P. Klas, N. Fuhr, B. Larsen, and A. Tombros. Designing a User Interface for Interactive Retrieval of Structured Documents Lessons Learned from the INEX Interactive Track. *Research and Advanced Technology for Digital Libraries*, pages 291–302, 2006.

S. Malik, A. Trotman, M. Lalmas, and N. Fuhr. Overview of INEX 2006. *Comparative Evaluation of XML Information Retrieval Systems*, pages 1–11, 2007.

K. Markey. Twenty-five years of end-user searching, Part 1: Research findings. *J. Am. Soc. Inf. Sci. Technol.*, 58(8):1071–1081, 2007.

V. Mihajlovic. *Score Region Algebra: A flexible framework for structured information retrieval*. PhD thesis, University of Twente, Enschede, December 2006.

V. Mihajlovic, H. E. Blok, D. Hiemstra, and P. M. G. Apers. Score region algebra: building a transparent XML-R database. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 12–19, New York, NY, USA, 2005. ACM.

D. R. Miller, T. Leek, and R. M. Schwartz. A hidden Markov model information retrieval system. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 214–221, New York, NY, USA, 1999. ACM.

G. Navarro and R. Baeza-Yates. A language for queries on structure and contents of textual databases. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 93–101, New York, NY, USA, 1995. ACM.

A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–266, New York, NY, USA, 2001. ACM.

R. O'Keefe and A. Trotman. The Simplest Query Language That Could Possibly Work. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*, 2003.

P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury. ORDPATHs: Insert-Friendly XML Node Labels. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France*, pages 903–908. ACM Press, 2004.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library, 1999.

C. R. Palmer, J. Pesenti, R. E. Valdés-Pérez, M. G. Christel, A. G. Hauptmann, D. Ng, and H. D. Wactlar. Demonstration of hierarchical document clustering of digital library retrieval results. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2001, Roanoke, Virginia, USA, June 24-28, 2001, Proceedings*, page 451. ACM, 2001.

D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão, editors, *CIKM*, pages 731–740. ACM, 2007.

D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal. Probabilistic question answering on the web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 408–419, New York, NY, USA, 2002. ACM.

G. Ramírez and A. P. de Vries. Combining Indexing Schemes to Accelerate Querying XML on Content and Structure. In V. Mihajlovic and D. Hiemstra, editors, *TDM*, CTIT Workshop Proceedings Series, pages 49–56. Centre for Telematics and Information Technology (CTIT), University of Twente, Enschede, The Netherlands, 2004.

M. Richardson and P. Domingos. The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *NIPS*, pages 1441–1448. MIT Press, 2001.

S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM.

H. Rode and D. Hiemstra. Using Query Profiles for Clarification. In M. Lalmas, A. MacFarlane, S. M. Rüger, A. Tombros, T. Tsikrika, and A. Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2006.

H. Rode and D. Hiemstra. Conceptual Language Models for Context-Aware Text Retrieval. In *Proceedings of the 13th Text REtrieval Conference Proceedings (TREC)*, 2004.

H. Rode, G. Ramírez, T. Westerveld, D. Hiemstra, and A. P. de Vries. The Lowlands' TREC Experiments 2005. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings the 14th Text REtrieval Conference (TREC)*, 2005.

H. Rode, P. Serdyukov, D. Hiemstra, and H. Zaragoza. Entity Ranking on Graphs: Studies on Expert Finding. Technical Report TR-CTIT-07-81, Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, November 2007.

H. Rode, P. Serdyukov, and D. Hiemstra. Combining Document- and Paragraph-Based Entity Ranking. In *Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*, 2008. to appear.

I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, 2003.

M. Rys. Full-Text Search with XQuery: A Status Report. In Blanken et al. (2003), pages 39–57.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297, 1990.

G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–58, New York, NY, USA, 1993. ACM.

S. Schlobach, D. Ahn, M. de Rijke, and V. Jijkoun. Data-driven Type Checking in Open Domain Question Answering. *J. of Applied Logic*, 5(1):121–143, 2007.

F. Sebastiani. Text Categorization. In A. Zanasi, editor, *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*, pages 109–129. WIT Press, Southampton, UK, 2005.

P. Serdyukov and D. Hiemstra. Modeling Documents as Mixtures of Persons for Expert Finding. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *ECIR*, volume 4956 of *Lecture Notes in Computer Science*, pages 309–320. Springer, 2008.

A. Shakery and C. Zhai. A probabilistic relevance propagation model for hypertext retrieval. In Yu et al. (2006), pages 550–558.

A. Sieg, B. Mobasher, and R. Burke. Inferring User's Information Context: Integrating User Profiles and Concept Hierarchies. In *Proceedings of the 2004 Meeting of the International Federation of Classification Societies*, Chicago, USA, 2004a.

A. Sieg, B. Mobasher, S. Lytinen, and R. Burke. Using Concept Hierarchies to Enhance User Queries in Web-based Information Retrieval. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, 2004b.

K. Spärck-Jones, S. E. Roberston, and M. Sanderson. Ambiguous requests: implications for retrieval tests, systems and theories. *SIGIR Forum*, 41(2): 8–17, 2007.

A. Spink, M. Park, B. J. Jansen, and J. Pedersen. Multitasking during Web search sessions. *Information Processing & Management*, 42(1):264–275, January 2006.

I. Tatarinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In *SIGMOD Conference*, pages 204–215. ACM, June 2002.

A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In Fuhr et al. (2005), pages 16–40.

T. Tsikrika, P. Serdyukov, H. Rode, T. Westerveld, R. B. N. Aly, D. Hiemstra, and A. V. de. Structured Document Retrieval, Multimedia Retrieval, and Entity Ranking Using PF/Tijah. In *Proceedings of the 6th Initiative on the Evaluation of XML Retrieval (INEX 2007), Dagstuhl, Germany*, Lecture Notes in Computer Science, pages 273–286, London, March 2008. Springer Verlag.

E. M. Voorhees and H. T. Dang. Overview of the TREC 2005 Question Answering Track. In *Proceedings the 14th Text REtrieval Conference (TREC)*, 2005.

F. Weigel, H. Meuss, F. Bry, and K. U. Schulz. Content-Aware DataGuides: Interleaving IR and DB Indexing Techniques for Efficient Retrieval of Textual XML Data. In S. McDonald and J. Tait, editors, *Advances in Information Retrieval, 26th European Conference on IR Research, ECIR 2004, Sunderland, UK, April 5-7, 2004, Proceedings*, pages 378–393. Springer, 2004.

R. Wilkinson. Effective retrieval of structured documents. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 311–317, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM.

P. S. Yu, V. J. Tsotras, E. A. Fox, and B. Liu, editors. *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, 2006. ACM.

H. Zaragoza, J. Atserias, M. Ciaramita, and G. Attardi. Semantically Annotated Snapshot of the English Wikipedia v.1 (SW1). `http://www.yr-bcn.es/semanticWikipedia`, 2007a.

H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi. Ranking very many typed entities on wikipedia. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 1015–1018, New York, NY, USA, 2007b. ACM.

C. Zhang, J. F. Naughton, D. J. DeWitt, Q. Luo, and G. Lohman. On supporting containment queries in relational database management systems. *SIGMOD Rec.*, 30(2):425–436, 2001.

J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 221–230, New York, NY, USA, 2007. ACM Press.

D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles. Co-ranking Authors and Documents in a Heterogeneous Network. In *ICDM*, pages 739–744. IEEE Computer Society, 2007.

J. Zhu, D. Song, S. Ruger, M. Eisenstadt, and E. Motta. The Open University at TREC 2006 Enterprise Track Expert Search Task. In *Proceedings the 15th Text REtrieval Conference (TREC)*, 2006.

J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6, 2006.

Summary

Text retrieval is an active area of research since decades. Several issues have been studied over the entire period, like the development of statistical models for the estimation of relevance, or the challenge to keep retrieval tasks efficient with ever growing text collections. Especially in the last decade, we have also seen a diversification of retrieval tasks. Passage or XML retrieval systems allow a more focused search. Question answering or expert search systems do not even return a ranked list of text units, but for instance persons with expertise on a given topic.

The sketched situation forms the starting point of this thesis, which presents a number of task-specific search solutions and tries to set them into more generic frameworks. In particular, we take a look at the three areas (1) context adaptivity of search, (2) efficient XML retrieval, and (3) entity ranking.

In the first case, we show how different types of context information can be incorporated in the retrieval of documents. When users are searching for information, the search task is typically part of a wider working process. This search context, however, is often not reflected by the few search keywords stated to the retrieval system, though it can contain valuable information for query refinement. We address with this work two research questions related to the aim of developing context-aware retrieval systems. First, we show how already available information about the user's context can be employed effectively to gain highly precise search results. Second, we investigate how such meta-data about the search context can be gathered. The proposed "query profiles" have a central role in the query refinement process. They automatically detect necessary context information and help the user to explicitly express context-dependent search constraints. The effectiveness of the approach is tested with retrieval experiments on newspaper data.

When documents are not regarded as a simple sequence of words, but

their content is structured in a machine readable form, it is attractive to try to develop retrieval systems that make use of the additional structure information. Structured retrieval first asks for the design of a suitable language that enables the user to express queries on content and structure. We investigate here existing query languages, whether and how they support the basic needs of structured querying. However, our main focus lies on the efficiency of structured retrieval systems. Conventional inverted indices for document retrieval systems are not suitable for maintaining structure indices. We identify base operations involved in the execution of structured queries and show how they can be supported by new indices and algorithms on a database system. Efficient query processing has to be concerned with the optimization of query plans as well. We investigate low-level query plans of physical database operators for the execution of simple query patterns. Furthermore, It is demonstrated how complex queries benefit from higher level query optimization.

New search tasks and interfaces for the presentation of search results, like faceted search applications, question answering, expert search, and automatic timeline construction, come with the need to rank entities instead of documents. By entities we mean unique (named) existences, such as persons, organizations or dates. Modern language processing tools are able to automatically detect and categorize named entities in large text collections. In order to estimate their relevance to a given search topic, we develop retrieval models for entities which are based on the relevance of texts that mention the entity. A graph-based relevance propagation framework is introduced for this purpose that enables to derive the relevance of entities. Several options for the modeling of entity containment graphs and different relevance propagation approaches are tested, demonstrating the usefulness of the graph-based ranking framework.

Samenvatting

Tekst-retrieval is sinds decennia een actief onderzoeksgebied. Meerdere onderwerpen zijn onderzocht gedurende dit tijdperk, zoals de ontwikkeling van statistische modellen voor de evaluatie van relevantie, of het efficiënt houden van retrieval ondanks steeds groeiende tekstbestanden. In de afgelopen tien jaar is bovendien een verdere diversificatie van retrievaltaken te herkennen. Passage- of XML retrievalsystemen maken het mogelijk het zoeken te beperken tot delen van de volledige tekst. "Question answering" of expertzoeksystemen leveren geen lijst van relevante documenten op maar bijvoorbeeld een lijst van personen met expertise in het genoemde vakgebied.

De beschreven situatie is het uitgangspunt van dit proefschrift, dat een aantal oplossingen voor specifieke zoektaken voorstelt en deze beschrijft in generiekere modellen. Met name onderzocht worden de gebieden: (1) context-specifiek zoeken, (2) efficiënte XML retrieval, en (3) het ordenen van entiteiten op relevantie.

In het eerst genoemde gebied laten we zien hoe verschillende soorten van contextinformatie gebruikt kunnen worden bij het zoeken naar documenten. Gebruikers zoeken meestal naar informatie in de context van een grotere taak. Deze context wordt echter zelden genoemd in het meestal beperkte aantal zoektermen in de query van een gebruiker, alhoewel de context vaak waardevolle informatie voor de inperking van een zoekopdracht bevat. We gaan in dit onderzoek vooral in op twee vragen, die bij de ontwikkeling van context-specifieke zoeksystemen een belangrijke rol spelen. Ten eerste laten we zien hoe beschikbare contextinformatie gebruikt kan worden voor het verbeteren van zoekresultaten. Ten tweede wordt onderzocht hoe zulke contextinformatie verzameld kan worden door het zoeksysteem. De voorgestelde "queryprofielen" spelen een centrale rol in het proces van het specificeren/beperken van de query. Ze helpen de noodzakelijke contextinformatie te herkennen en steunen de gebruiker bij het beperken van de

query. De effectiviteit van de aanpak is getest op de geselecteerde dimensies van contextinformatie.

Als documenten niet meer beschouwd worden als een simpele sequentie van woorden, maar hun inhoud gestructureerd is in een voor een machine leesbare vorm, is het aantrekkelijk retrieval systemen te ontwerpen die geschikt zijn voor de omgang met de toegevoegde structuurinformatie. Een voorwaarde voor structuur-retrieval is het ontwerp van vraagtalen die het voor de gebruiker mogelijk maken een inhouds- en structuurvraag te specificeren. Van bestaande vraagtalen wordt hier onderzocht in hoeverre ze de fundamentele eisen van structuur-retrieval steunen. Het onderzoek betreft echter vooral de efficientie van XML- of structuur-retrievalsystemen. Conventionele geïnverteerde indices voor documentretrieval zijn niet geschikt voor het opslaan van gestructureerde documenten. Het voorliggend onderzoek identificeert basisoperatoren bij de uitvoering van gestructureerde queries en laat zien hoe deze kunnen worden ondersteund door nieuwe indices en specifieke algoritmen draaiend op een database systeem. Efficiënte queryverwerking is ook gebaat bij de optimalisatie van queryplannen. We onderzoeken hier eerst queryplannen van database operatoren voor simpele patronen van queries. Later is ook gedemonstreerd dat queryplanoptimalisatie op hoger niveau helpt de uitvoeringstijd van complexe queries te verkorten.

Nieuwe zoektaken en gebruikersinterfaces, zoals “faceted search” applicaties, “question answering”, expertzoeksystemen, of de automatische generatie van onderwerp-gerelateerde tijdlijnen, vragen om het ordenen van entiteiten in plaats van het ordenen van documenten. Met entiteiten bedoelen we unieke, benoemde existenties zoals personen, organisaties of data. Nieuwe taalverwerkings- en herkenningssoftware kan entiteiten in grote tekstbestanden automatisch herkennen en categoriseren. We ontwerpen in dit proefschrift retrievalmodellen voor het ordenen van entiteiten met hulp van de relevantie van teksten die de entiteiten noemen. Een graaf-gebaseerd raamwerk wordt voorgesteld voor het verspreiden van relevantie in een graaf. Met behulp van dit raamwerk kan de relevantie van entiteiten worden afgeleid. Meerdere modeleringsmogelijkheden voor zogenaamde “entity containment” grafen en verschillede relevantie-verspreidingsmodellen zijn getest en laten het voordeel van het graaf-gebaseerde raamwerk zien.

SIKS Dissertation Series

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modeling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)

Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coupé (EUR)
Sensitivity Analyis of Decision-Theoretic Networks

2000-9 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

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 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 Petkovic (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 politiële 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 Smiljanic (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 Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval

2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mika (VU)
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)

Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic' (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Niek Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)
On the development an management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramírez Camps (CWI)
Structural Features in XML Retrieval

2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement

2008-01 Katalin Boer-Sorbán (EUR)

Agent-Based Simulation of Financial Markets: A modular, continuous-time approach

2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations

2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU)
Supporting the tutor in the design and support of adaptive e-learning

2008-08 Janneke Bolt (UU)
Bayesian Networks: Aspects of Approximate Inference

2008-09 Christof van Nimwegen (UU)
The paradox of the guided user: assistance can be counter-effective

2008-10 Wouter Bosma (UT)
Discourse oriented summarization

2008-11 Vera Kartseva (VU)
Designing Controls for Network Organizations: A Value-Based Approach

2008-12 Jozsef Farkas (RUN)
A Semiotically Oriented Cognitive Model of Knowledge Representation

2008-13 Caterina Carraciolo (UVA)
Topic Driven Access to Scientific Handbooks

2008-14 Arthur van Bunningen (UT)
Context-Aware Querying; Better Answers with Less Effort

2008-15 Martijn van Otterlo (UT)
The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.

2008-16 Henriëtte van Vugt (VU)
Embodied agents from a user's perspective

2008-17 Martin Op 't Land (TUD)
Applying Architecture and Ontology to the Splitting and Allying of Enterprises

2008-18 Guido de Croon (UM)
Adaptive Active Vision

2008-19 Henning Rode (UT)
From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search

2008-20 Rex Arendsen (UVA)
Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven.

2008-21 Krisztian Balog (UVA)
People Search in the Enterprise