DISCOVERING AND USING IMPLICIT DATA FOR INFORMATION RETRIEVAL

A Dissertation Presented

by

XING YI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2011

Computer Science

DISCOVERING AND USING IMPLICIT DATA FOR INFORMATION RETRIEVAL

A Dissertation Presented

by

XING YI

Approved as to style and content by:

James Allan, Chair

W. Bruce Croft, Member

David D. Jensen, Member

Michael Lavine, Member

Andrew G. Barto, Department Chair
Computer Science

Dedicated to my parents
Shuilian Huang
Jinrong Yi

ACKNOWLEDGMENTS

This thesis would not have been possible without the help of my mentors, colleagues and friends at the University of Massachusetts Amherst (UMass).

First and foremost, I would like to thank my advisor Prof. James Allan for providing both intellectual guidance and financial support for me during my graduate school life. Professor Allan is an excellent mentor who has always provided insightful advice and timely assistance for my research and study. I have learned many valuable lessons in how to choose research topics, how to explore research problems and develop possible solutions, and how to tackle research obstacles from him. These fortunes will accompany with me through my future scientific career. Prof. Allan has always provided valuable suggestions and timely feedback on my work and steered me in the correct direction when I felt lost in my research. He has often reminded me that as a researcher we should always try to think deeply, broadly, and at a high point of view, so that we can see the wood behind the trees instead of being lost in the details of models and algorithms. He also taught me how to discover the intrinsic and essential nature of different research problems. I am deeply grateful to Prof. Allan for his guidance as well as for his encouragement during my most difficult time. I will also be forever grateful to Prof. Allan for his great advice on writing papers and making presentations.

I would like to thank Prof. Bruce Croft for his assistance and advice during my time as a graduate student in the Center for Intelligent Information Retrieval (CIIR). I greatly enjoyed his textbook on Information Retrieval (IR) research, his seminar on Query Modeling and Understanding and his keynote talks in many conferences. I have learned a lot from his knowledgable comments and insightful discussions in our

CIIR internal lab meetings. Similar to all the other CIIR's students, I have greatly benefitted from Prof. Croft's influence on and leadership of the CIIR. I am also grateful to Prof. Croft for his important suggestions on my thesis proposal: these suggestions help me create the layout and the content of this thesis. I also would like to thank him for going through this thesis with great care and patience.

I would like to thank my other dissertation committee members, Professors David Jensen and Michael Lavine, for their time and their valuable insights and feedback.

Many of the ideas for this thesis stemmed from the cooperative research during 2005–2007 with Dr. Victor Lavrenko on addressing the incomplete/empty field problem encountered when searching semi-structured documents. I greatly enjoyed the interactions and discussions with him. Before he left the CIIR in 2007, he gave me some valuable advice on plausible research directions to continue our work. The major content of Chapter 2 of this thesis comes from our cooperative research. I would like to thank Dr. Lavrenko for his important contributions to this thesis research.

I would like to thank my mentor Dr. Chris Leggetter in Yahoo! labs and my previous colleague Dr. Hema Raghavan for their help in our cooperative research on discovering web users' implicit geographic web search intents. I greatly enjoyed our weekly meetings and discussions. The discovery of this research forms the major part of Chapter 5 of this thesis.

I would like to thank my Master thesis advisor Prof. Changshui Zhang in Tsinghua University for initially leading me into the road of doing research in computer science. He opened the door of Machine Learning and Artificial Intelligence research area for me. My master thesis in Tsinghua University is to address missing label problems in Machine Learning tasks. The influence of his early guidance is also reflected in this thesis research which addresses missing data issues in many real-world IR tasks.

I would like to give my special and deep thanks to my best friend forever, Dr. Xuan Huang, for her continuous support during my graduate school student life in

UMass. Without Dr. Huang being aside, going through this journey would have been much harder if not impossible.

I would like to thank all of my fellow students and colleagues at the CIIR for making our lab a great place to doing research in IR. Every student specializes in a different IR research direction and has expertise in different research topics. This situation results in a highly cooperative and productive research environment, where we frequently discuss frontier IR research topics, participate creative brain-storms and give constructive feedback for each other's work. I feel particularly lucky that my time in the CIIR straddles a period when one large group of fellow students are leaving and another large group of students are joining the lab; therefore, I have the chance to work with people that have a great variety of expertise in IR. To the former group of my fellow students, in particular, I would like to thank Hema Raghavan, Fernando Diaz, Ao Feng, Yun Zhou, Xing Wei and Shaolei Feng for their friendship and for their great conversations about research and life. I thank Giridhar Kumaran's help for the GALE project, Mark Smucker's help for the InterNano project, Ben Carterette's help for the TREC Million Query Tracks, Trevor Strohman and Donald Metzler's help for my getting familiar with the Lemur/Indri search systems. To the latter group of my fellow students, in particular, I would like to thank Xiaobing Xue, Elif Aktolga, Jeffery Dalton, Niranjan Balasubramanian, Jangwon Seo, Jinyoung Kim, Michael Bendersky and Van Dang for their friendship and for their cooperation and discussions. I thank Henry Feild's help for the UptoDate Project, Marc Cartright and Jeffery Dalton's help for the Megabook Project, Samuel Huston's help for getting familiar with the Galago search system. There are many more CIIR fellow students and I would like to thank them all for their help for my research and work.

I would like to thank the excellent staff of the CIIR and the computer science department. David Fisher must be especially thanked for his help for addressing my questions/bugs encountered when I used the Lemur/Indri search systems. I greatly

appreciate all the assistance and help from Kate Moruzzi. I would like to specially thank Valerie Caro and Daniel Parker for their help for my using the Swarm and Sydney Grid systems for experiments. Thanks also to Andre Gauthier, Jean Joyce and Glenn Stowell for their timely aid whenever needed.

Obtaining the Ph.D. degree means my school life is past and a new journey begins. To be able to reach this point, I owe much credit to my parents. They have made numerous sacrifices to ensure that I have a good life and a great learning environment. They have always encouraged me to pursue for knowledge since I was a child. Even I have to study abroad and be far away from home, they have given all their support and have never complained a word although I am their single child. I owe everything I am to them. My parents have always told me that the most important achievement is not the result but the procedure. I will keep their advice in mind and always try my best in my future career.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advanced Research Projects Agency (DARPA) under contract #HR0011-06-C-0023, in part by NSF grant #IIS-0910884 , in part by Monster, and in part by UpToDate . Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of the sponsor.

ABSTRACT

DISCOVERING AND USING IMPLICIT DATA FOR INFORMATION RETRIEVAL

SEPTEMBER 2011

XING YI

B.Eng., TSINGHUA UNIVERSITY, BEIJING, CHINA

M.Eng., TSINGHUA UNIVERSITY, BEIJING, CHINA

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed by: Professor James Allan

In real-world information retrieval (IR) tasks, the searched items and/or the users' queries often have *implicit* information associated with them – information that describes unspecified aspects of the items or queries. For example, in web search tasks, web pages are often pointed to by hyperlinks (known as anchors) from other pages, and thus have human-generated succinct descriptions of their content (anchor text) associated with them. This indirectly available information has been shown to improve search effectiveness for different retrieval tasks. However, in many real-world IR challenges this information is *sparse* in the data; i.e., it is *incomplete or missing in a large portion of the data*. In this work, we explore how to discover and use implicit information in large amounts of data in the context of IR.

We present a general perspective for discovering implicit information and demonstrate how to use the discovered data in four specific IR challenges: (1) finding rele-

vant records in semi-structured databases where many records contain incomplete or empty fields; (2) searching web pages that have little or no associated anchor text; (3) using click-through records in web query logs to help search pages that have no or very few clicks; and (4) discovering plausible geographic locations for web queries that contain no explicit geographic information.

The intuition behind our approach is that *data similar in some aspects are often similar in other aspects*. Thus we can (a) use the observed information of queries/documents to find similar queries/documents, and then (b) utilize those similar queries/documents to reconstruct *plausible* implicit information for the original queries/documents. We develop language modeling based techniques to effectively use content similarity among data for our work. Using the four different search tasks on large-scale noisy datasets, we empirically demonstrate the effectiveness of our approach. We further discuss the advantages and weaknesses of two complementary approaches within our general perspective of handling implicit information for retrieval purpose.

Taken together, we describe a general perspective that uses contextual similarity among data to discover implicit information for IR challenges. Using this general perspective, we formally present two language modeling based information discovery approaches. We empirically evaluate our approaches using different IR challenges. Our research shows that supporting information discovery tailored to different search tasks can enhance IR systems' search performance and improve users' search experience.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

In many information retrieval (IR) tasks, the searched items and/or the users' queries often have associated *implicit* information – information that describes unspecified aspects of the items or queries. For example, websites such as Monster Worldwide [1] and Linkedin [2] ask users to provide their personal information using online forms that contain many textual fields instead of using unstructured plain text boxes. Those textual fields can usually provide important information of users' profiles such as users' skills, education, work experience, etc. However, users tend to fill the fields carelessly and thus leave many incomplete or empty fields. In many cases, although the information is not *explicitly* present in some fields, it is implicit in other fields; e.g., specific skills might be omitted from a user profile, but are implied by – so implicit in – the work experience.

As another example, many IR tasks such as Google blog search and Monster job search allow users to formulate advanced queries that contain multiple textual fields to better represent their information need. The fields in those complex queries provide useful information to help searching relevant items for the users. Again, users often include only some of them and omit most fields, but often the information in those empty query fields is implicit in other fields – e.g. the book search query {*title* = "differential geometry", *subject* = "math" *audience level*=" "} (i.e. the *audience*

[1] `http://www.monster.com`

[2] `http://www.linkedin.com/`

level field is empty) strongly suggests that the target audience level of the relevant books may be *undergraduate* or higher.

The additional aspects of the data – the searched items and queries – are usually very helpful for finding relevant information and play an important role in designing and enhancing modern IR systems for different retrieval tasks. Unfortunately, previous research has shown that the examples above are realistic: human-generated information is often *sparse* in the data, i.e. it is *incomplete or missing in a large portion of the data*. This situation presents a major obstacle to many existing retrieval techniques that leverage this information for improving search. Here are more examples from real-world search tasks:

1. In a 2004 snapshot of the National Science Digital Library collection, only 3.5% of the records mention the target audience. Thus, if a query contains *audience = 'elementary school'*, it will consider at most 3.5% of all resources in the collection when the simple exact-match approach is used. The audience field's value is missing, so is at best implicit information in 96.5% of the records.

2. A traveler may issue the query 'space needle tour' to web search engines without explicitly specifying the geographic location '*seattle*' in the query. Search engines that leverage explicit geographic information to provide relevant travel information will not work for this query. The intended location of the query is implicit information.

3. In email search tasks, users may specify keywords in the subject field to search previously received emails; however, the target emails may have subject keywords different from the user-specified ones, or even worse, only have empty or meaningless subject fields (such as ones only containing 'Re:' or 'No subject'). The actual subject of a message is frequently implicit or inaccurate information.

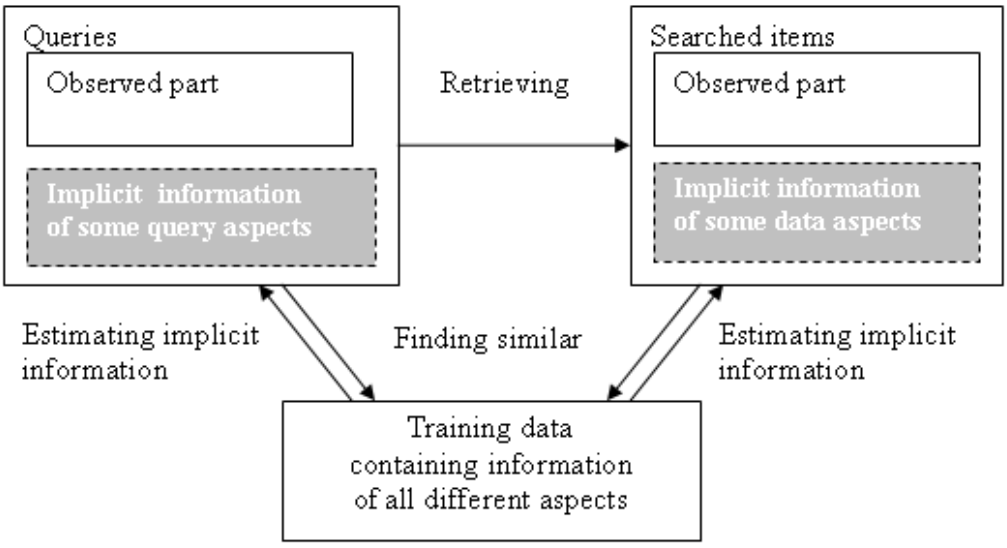


Figure 1.1. The general perspective of our implicit information discovery approach in an IR context

4. In some online product search tasks, potential buyers may query with incorrect or missing information for some key attributes (such as brand names or sizes) of the products they are interested in. Thus, the search engines of web companies cannot match relevant products to the buyers' request. The brand name or size is implicit information here.

Ignoring the implicit information in the above examples will degrade the retrieval performance of the IR systems and negatively affect users' search experience. In this research, we are concerned with methods for discovering and using implicit data aspects for retrieval purpose.

Our approach is illustrated by the high-level general perspective depicted in Figure 1.1. The upper-left and upper-right external boxes denote the information that consists of different data aspects on the query side and the searched item side, respectively. The implicit data aspect information of the queries or the searched items is denoted by two shadowed internal boxes surrounded by the dashed lines in the figure.

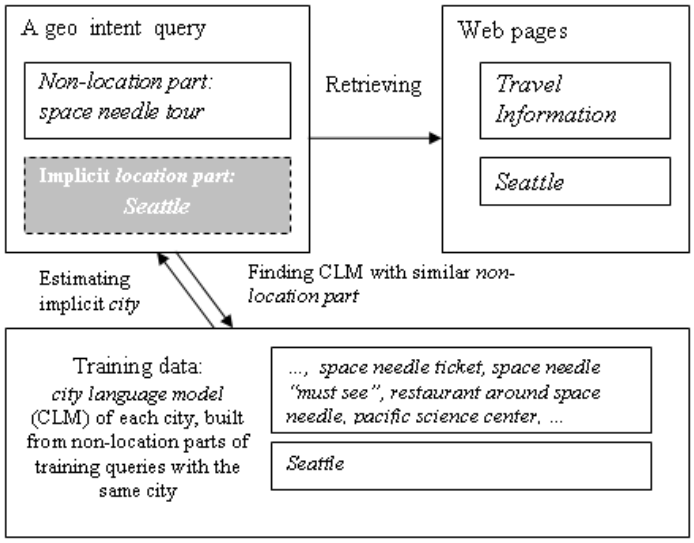


Figure 1.2. Illustration of our approach of discovering implicit geographic information for a query: "space needle tour" for web search.

The bottom box denotes available training data (queries or searched items depending on the search task) that contain observed information for all different aspects. Our goal is to more effectively retrieve items that are relevant to users' information needs by discovering plausible implicit data aspects on the query side or on the searched item side (depending on the search task).

Our approach consists of three basic steps: (1) use the observed part (the unshadowed internal boxes in Figure 1.1) of the queries or searched items to find similar training data; (2) use the observed information in the similar training data to estimate plausible implicit information for the corresponding data aspects of the queries or searched items; (3) use the original and the reconstructed data aspect information for retrieving relevant items. Our approach takes advantage of the fact that *data similar in some aspects are often similar in other aspects.*

As an example, Figure 1.2 shows intuitively how we use this general perspective to discover implicit geographic information in the second example discussed above (in page 2). In this figure, we (1) use the observed non-location part ("space needle tour")

of the query to find a city language model that has similar non-location information in the training data; (2) then we can predict the likely related city ("Seattle") for the query; in the end, (3) we can provide more travel information around Seattle for the traveler.

In this thesis, based on the general perspective in Figure 1.1, we develop two complementary language modeling based approaches – the *query-side* and the *searched-item-side* approaches – for discovering and using implicit data information for different real-world IR challenges.

The query-side approach uses our developed technique called *Structured Relevance Models* (SRM), and is depicted on the left side or the query side in Figure 1.1. In this SRM based approach, we (a) design probabilistic generative language models to infer plausible (but missing) information for the observed queries and then (b) search plausibly relevant items that can match the original and/or the new discovered query aspects. We will formally describe the SRM technique in §2.3 of Chapter 2. Using a hypothetical probabilistic model of generating semi-structured data (queries/documents), SRM can leverage the dependencies of the words (or other discrete attribute values) within and across data aspects for effectively discovering implicit information in different data aspects that use very different languages. Thus, the query-side approach can find relevant items that cannot be retrieved by only using the original query, by extending the query to cover every aspect of searched items.

The searched-item-side approach follows a *contextual language translation* (CLX) approach (Jing and Croft 1994; Xu and Croft 1996; Wang and Zhai 2008) and is depicted on the right side or the searched-item side in Figure 1.1. Here, we (a) use the CLX approach to infer plausible implicit information for a small set of items (retrieved using the original queries) from the items' observed part and then (b) *rerank* the items based on both the original and the discovered data aspects of the items, in order to push the highly relevant ones up to the top of the ranked list. We will formally describe

our CLX based approach used for the searched-item-side information discovery in §3.3.2 of Chapter 3 and §4.3.2 of Chapter 4. The CLX approach leverages rich textual content of the search-items to reliably infer implicit information for different data aspects of the search-items. Then the discovered information is used to smooth the content of the search-items for reranking. Therefore, compared with the query-side approach that uses the discovered information for query expansion, the searched-item-side approach is more resistant to irrelevant noise from the training data (depicted in the bottom box in Figure 1.1) and topic-drifting. Nevertheless, this approach assumes that the implicit data aspects share some vocabulary with the content of the searched items and the queries; in addition, it uses a reranking scheme which relies on the quality of the top ranked items returned by the original query. Both issues limit the usage of this approach.

This thesis will demonstrate how we can use our general perspective in Figure 1.1 and its two approaches above to discover implicit information for different IR tasks. To limit the scope of our discussion and illustrate how our approach works in practice, we focus on four specific real-world IR challenges: (1) finding relevant records in semi-structured databases where many records contain incomplete or empty fields; (2) searching web pages that have little or no associated anchor text; (3) using click-through information in web query logs to help search pages that have no or very few clicks; and (4) discovering plausible city information for web queries that contain no explicit geographic information.

Next, we provide a general brief discussion of previous work related to our research in §1.1. Then in §1.2 to §1.5, we use the general perspective in Figure 1.1 to introduce each of the above IR challenges to be addressed in this thesis. After that we summarize the general contributions of our research in §1.6 and present the structure of the remaining chapters of the thesis in §1.7.

1.1 Related Information Discovery

1.1.1 Missing Data Issue in Statistical Learning

Because we specifically focus on discovering implicit data in the context of IR in this research, it is worthwhile to briefly review two missing data mechanism assumptions – missing completely at random (MCAR) and missing at random (MAR) – which are well known in the statistics and machine learning research community when using incomplete data for learning tasks (Koller and Friedman 2010, pp.850–856). Intuitively, the MCAR assumption is that the missing data mechanism is *completely independent* of the domain variables thus the missing values are randomly distributed across all observations. The MAR assumption has a weaker assumption on the missing data mechanism: it assumes that this mechanism is *conditionally independent* of the missing values given the observed variables; thus, the mechanism *does not* depend on the true value of the missing variable, but it may depend on the value of other observed variables. In our research, where we need to handle implicit data in real-world searched item collections and users' queries, we usually face a more complex situation, where data are hidden for greatly varied reasons and that both MCAR and MAR are often violated. Therefore, we do not make general missing data assumptions for all different IR challenges, but describe the individual implicit data aspect addressed in each retrieval task instead.

To handle missing data problems and also learn probabilistic models that have hidden variables, a parametric approach – the Expectation-Maximization (EM) algorithm – has been widely used in the statistics and machine learning research (Dempster *et al.* 1977; Little and Rubin 1986), where the missing data and the probabilistic model parameters are estimated and updated iteratively to maximize log-likelihood of the data. In our research, we focus on a common missing information situation in the IR context, where plausible implicit values are from a large vocabulary of natural language. Different from the parametric approach, we develop

language modeling based *non-parametric* approaches to estimate implicit information for queries and/or searched items based on their observed information (Lavrenko 2004; Silverman 1986). Based on the assumption that data similar in some aspects are often similar in other aspects, our approach uses contextual language similarity for effectively discovering implicit aspects of queries/searched items for retrieval. We point out that here we do not model the missing data mechanism (e.g. MCAR, MAR) in our implicit information discovery approach. We leave approaches that explicitly model the missing data mechanism as future work.

1.1.2 Information Discovery in Information Retrieval

Because user-specified query information is often not directly available (thus *implicit*) in relevant documents, one core part of IR research is investigating how to bridge the semantic gap between the users' input queries and their relevant documents, in order to better search relevant information for the users. To achieve this goal, a lot of research has been devoted into two major directions: (1) leveraging all available information (explicit and/or implicit) to reformulate the user-specified queries for better representing users' information need and matching relevant documents; and (2) enriching the representation of each document and inferring each document's implicit information for better matching related queries. Both research directions cover many important IR research issues. The goal of our thesis research is to discover implicit information for different aspects of queries/searched items, in order to help bridge the vocabulary gap between user-specified query information and relevant items. Therefore our research contributes to this important IR research area of addressing the semantic gap challenge in both research directions. Next, we will name a few important research issues that are closely related to our research in each direction. Our intent is to provide a high-level picture of how the thesis research

contributes to this IR research area, rather than an exhaustive list of research issues in this area. More detailed work is provided in the individual chapters as needed.

In the first research direction (reducing the semantic gap from the query side), one important research issue is to reduce *vocabulary mismatch* through *query expansion*. The vocabulary mismatch problem is one of the major causes of failures in IR systems and happens because users often describe their information need using different words than are found in relevant documents (Croft 1995). To address this problem, many effective automatic query expansion techniques have been designed (Rocchio 1971; Robertson 1991; Lavrenko and Croft 2001; Zhai and Lafferty 2001a) for discovering plausibly useful terms that can help to identify relevant documents from either top-ranked documents initially retrieved using the original query (pseudo-relevance feedback) or judged relevant documents (relevance feedback). The discovered terms (most of which are missing thus *implicit* in the original query) can usually help find more relevant documents and greatly improve search performance in many search tasks, thus effectively reducing vocabulary mismatch. Our SRM based query-side information discovery approach (depicted on the left side of Figure 1.1) directly extends *relevance-based language models* (Lavrenko and Croft 2001), a highly effective version of the above query expansion techniques, to discover implicit information for different query aspects for better representing users' information need and reducing vocabulary mismatch. Different from the above classical query expansion techniques which usually focus on handling *unstructured* plain-text queries, our approach considers more complex retrieval scenarios where different query aspects are used for search and each query aspect may contain implicit information represented by a very different language.

Another important research issue in addressing the semantic gap from the query side is to discover queries' inherent semantic structure, which is often not explicitly presented (e.g. when queries are input through an *unstructured* plain-text search

box), for more accurately representing users' information need. Then the discovered semantic units in the queries can be used for matching the corresponding semantic units (which may or may not be explicit) in documents to precisely search relevant documents. These semantic units could be named entities, concepts (noun phrases), n-gram phrases, semi-structured fields as well as other information units that represent certain unspecified aspects of users' information need. On this research issue, Metzler and Croft (2005) developed a Markov random field model based general framework to use term dependencies (including ordered/unordered phrases and other term proximity information) for better searching relevant documents; they further proposed (2007) using their model to discover latent concepts from pseudo-relevant or relevant documents for query expansion. Bendersky and Croft (2008a) proposed using a supervised machine learning technique for discovering key concepts in plain-text verbose queries and re-weighting these concepts in retrieval models to achieve better search performance. Guo et al. (2008) developed a unified model based on the Conditional Random Field technique for simultaneously predicting hidden phrasal structure and correcting possibly existing errors for queries. Kim et al. (2009) proposed a language modeling based approach to discover implicit semi-structured field structure in unstructured plain-text queries for helping search semi-structured documents. The above research complements our research: we assume that queries' semantic structures (or data aspects) are known beforehand or have been discovered using schemes from the above research and focus on discovering implicit information in different query aspect for helping search.

In the second research direction (reducing the semantic gap from the document side), similarly, one important research issue is to reduce *vocabulary mismatch* through *document expansion*. Document expansion techniques typically enrich each document's content using words from its similar documents (Kurland and Lee 2004; Liu and Croft 2004; Tao *et al.* 2006; Mei *et al.* 2008) or from the document's la-

tent topics which are discovered by applying statistical topic models on the searched collection (Hofmann 1999; Wei and Croft 2006; Yi and Allan 2009). In this way, each query can be better covered by the enriched content of its plausibly relevant documents. The above research has shown that similar to query expansion, the document expansion approach can also statistically significantly improve search performance and effectively reduce vocabulary mismatch. Similar to some of the above document expansion techniques that infer each document's implicit content from its similar documents, our CLX based searched-side approach (depicted on the right side of Figure 1.1) infers implicit information of each different data aspect of a searched item from the corresponding data aspect of its similar items. Different from the typical document expansion approach that usually focuses on enriching the content representation of documents, our approach considers the situation where some unspecified aspects of searched items besides their content can be used for search and each data aspect may contain implicit information that can be inferred from the observed other aspects of each item.

Recently, IR researchers have begun to address the data sparseness issue that exists in many real-world IR tasks, such as web search (Craswell and Szummer 2007; Gao *et al.* 2009; Metzler *et al.* 2009; Seo *et al.* 2011) and collaborative filtering (Ma *et al.* 2007), in order to further improve retrieval effectiveness. As mentioned in the introduction of this chapter, although human-generated information is usually highly effective for helping search and reducing semantic gap between queries and relevant documents, it is often very sparse. Thus, it is unreliable to directly use this information for retrieval. Researchers have explored how to reduce data sparseness using different available information in different specific search tasks. To address *anchor text sparsity* for web search, Metzler et al. (2009) proposed using the web hyperlink graph and propagating anchor text over the web graph to discover missing anchor text for web pages. To address *click-through data sparseness* for web search,

Craswell and Szummer (2007) proposed applying Markov random walk algorithm on the query-URL click graph to find plausible missing clicks; Gao et al. (2009) proposed a Good-Turing estimator (Good 1953) based method to smooth click-through features for web pages that have received no clicks; Seo et al. (2011) proposed two techniques for smoothing click counts based on a statistical model and spectral analysis of document similarity graph. To address *user-item rating sparseness* for collaborative, Ma et al. (2007) proposed estimating the missing rating of an item from a user by averaging ratings from similar items and similar users[3]. Our research also addresses data sparseness for some of the above IR tasks, but we focus on discovering implicit language information for different query/searched-item aspects and reduce data sparseness following the formal language modeling retrieval framework (Ponte and Croft 1998).

To summarize, this thesis research directly relates to and contributes to a classical and core IR research area of bridging the semantic gap between users-specified queries and relevant items, and also an important frontier IR research area of addressing the data sparseness issue for many real-world IR tasks where human-generated information is used for improving search performance.

1.2 Discovering Implicit Field Values for Searching Semi-structured Records

The use of semi-structured documents, such as HTML/XML documents, to store information and data has been quickly expanding. This trend will presumably continue due to the convenience of using semantic document structures to represent human knowledge. Using a traditional relational database and the Structured Query Language (SQL) keyword-match approach to search semi-structured data runs into a

[3]The similarity between two items/users is measured by the correlation of their ratings from/over the same set of users/items, respectively.

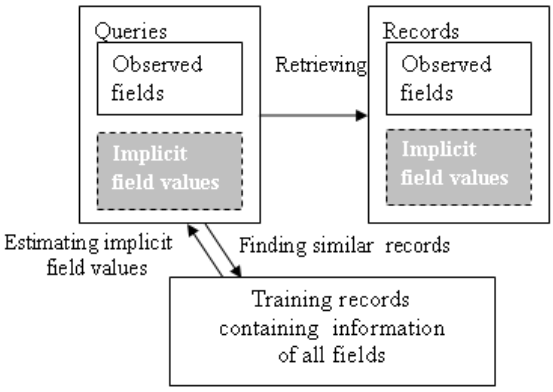


Figure 1.3. The specific perspective of discovering implicit field values for searching semi-structured records

number of obstacles: inconsistent schemata (e.g. different markups that represent the same semantic units), unstructured natural language fields, and even empty fields.

When searching semi-structured records, both the records and the queries may have incomplete or empty fields; the original user-specified query fields may be partially or completely missing in the search target collection. To address this issue, we use the approach depicted in Figure 1.3 to discover the implicit field values for search, following the SRM based query-side approach from our general perspective in Figure 1.1.

Here the searched items are semi-structured records and the implicit data information is in semi-structured fields. We hypothesize that *semi-structured records that have similar attribute values in some fields may have similar attribute values in other fields* due to the cross-field relations between attribute values in different fields. For example, articles with "quantum" in their titles are more likely to share a higher reading level. Using this assumption, we discover plausible implicit field values of semi-structured records by using their similar records' corresponding information. As shown in Figure 1.3, given a query, we first leverage the observed fields of the query to find training records that have similar fields; then we use the infor-

mation in the similar training records to estimate an extended semi-structured query that covers all record fields in the searched collection. Each field in this extended query now contains plausible field values indicated by the original query. Finally, all the records are ranked by their language modeling based similarity to the extended query. Here we only consider the SRM based query-side approach instead of the CLX based searched-item-side approach that uses reranking, because that the query-side approach can better handle the situations that (1) different fields often use very different languages and (2) the original query fields may be completely missing in many relevant records.

We use the SRM based approach of Figure 1.3 to address two large-scale real-world semi-structured record searching tasks in Chapter 2. The first is to find relevant records in the National Science Digital Library (NSDL) record collection. The second is to match semi-structured job and resume electronic records in a industry-scale job/resume collection provided by Monster Worldwide, a well-known online job service company.

1.3 Discovering Implicit Anchor Text Information for Web Search

There are rich dynamic human-generated hyperlink structures on the web. Most web pages contain some hyperlinks, referred to as *anchors*, which point to other pages. Each anchor consists of a destination URL and a short piece of text, called *anchor text*. Anchors play an important role in helping web users conveniently navigate the web for information they are interested in, in part because anchor text usually provides a succinct description of the destination URL's page content. The description means that anchor text is very helpful for web search. However, most web pages have few or no incoming hyperlinks (anchors) and therefore lack associated anchor text information (Broder *et al.* 2000). This situation is known as the *anchor text*

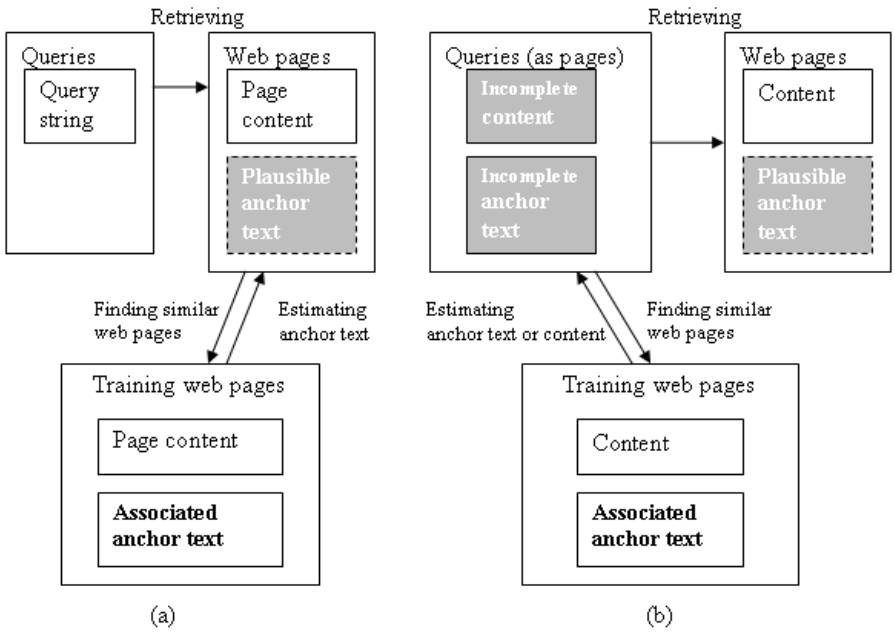


Figure 1.4. The specific perspective of discovering anchor text for web search: (a) using similar web pages for anchor text discovery; (b) viewing queries as web pages and reconstructing better queries for search.

sparsity problem (Metzler *et al.* 2009) and presents a major obstacle for any web search algorithms that want to use anchor text to improve retrieval effectiveness.

We use both the query-side and the searched-item-side approaches depicted in Figure 1.4 to address the above anchor text sparsity problem, following the general perspective from Figure 1.1. Here, the searched items are web pages and the implicit information of data is the web pages' associated anchor text. We hypothesize that *web pages that are similar in content may be pointed to by anchors having similar anchor text* due to the common semantic relation between anchor text and page content. Under this assumption, the two approaches in Figure 1.4 use the similarity among web pages and their anchor text to discover plausible anchor text information for web search. These approaches are briefly described as follows.

In the CLX based searched-item-side approach, shown in Figure 1.4(a), we run each original web query, which is an *unstructured* plain text string, against the web

page collection to retrieve a small subset of web pages that may be relevant to the query. Next, for each page in the retrieved set, we discover plausible anchor text and then use the page content and the discovered anchor text information together to *rerank* the page. To discover a page's implicit anchor text, we first find training web pages similar in content to the target page, then use those pages' associated anchor text to estimate the plausible anchor text for the page. The whole web page collection and all anchor text in the collection are used as the training data.

In the SRM based query-side approach, shown in Figure 1.4(b), we add structure to the original unstructured web queries and adapt the approach in §1.2 (depicted in Figure 1.3) for search. The basic idea of this approach is to first discover implicit information in the (now structured) queries and then search with the extended queries. We view queries as very short web pages that contain two fields: *Content* and *Associated Anchor Text*. The two shadowed internal boxes in the upper-left external box (query side) in the figure are surrounded by solid lines because we assume that they are observed but incomplete. We also assume that both the observed *Content* and *Associated Anchor Text* fields contain the same copy of the original query string: intuitively, the query is searching for pages that match the query in content and/or anchor text. We first leverage the observed fields of a query to find training pages that have similar fields. After that, we use the information in the similar training pages to estimate all plausible implicit field values indicated by the original query. Finally, all the pages to be searched will be ranked by their language modeling based similarity to the extended query.

In Chapter 3, we use the two approaches above to address the anchor text sparsity problem for the standard TREC web search tasks.

1.4 Discovering Missing Click-through Information for Web Search

The click-through information in web search query logs contains important user preference information (both individual and collective) over the returned web search results. This information plays an important role in designing and enhancing modern web search engines. However, click-through data usually suffer from a data sparseness problem where a large volume of queries have few or no associated clicks. This is known as the *missing click problem* or *incomplete click problem* in web search (Gao *et al.* 2009).

We employ both the query-side and the searched-item-side approaches depicted in Figure 1.5 to address the above missing/incomplete click problems, again following the general perspective from Figure 1.1. Note that these two approaches are similar to those for addressing the anchor text sparsity problem in §1.3. Here, the searched items are web pages and the implicit information is web pages' *click-associated* queries (i.e. queries that led to clicks on the pages). We hypothesize that *web pages that are similar in content may be clicked by web searchers issuing similar queries*, because of the semantic relation between queries and the web page content of their clicked URLs. Under this assumption, the two approaches in Figure 1.5 use the semantic similarity among web pages and their click-associated queries to discover plausible click-through query language information for helping search. These approaches are briefly described as follows.

In the CLX based searched-item-side approach, shown in Figure 1.5(a), we run each original web query, which is an *unstructured* plain text string, against the web page collection to retrieve a small subset of web pages that may be relevant to the query. Next, for each page in the retrieved set, we discover the page's plausible click-associated queries and then use the page content and the discovered query content together to *rerank* the page. To discover a page's click-associated information, we

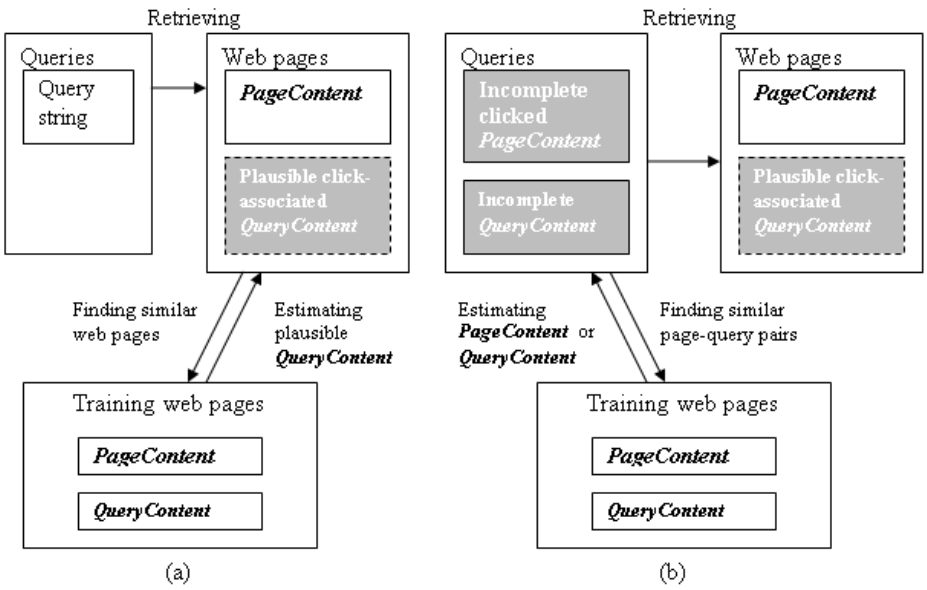


Figure 1.5. The specific perspective of discovering plausible click-through features for web search: (a) using similar web pages for discovering plausible click-associated queries; (b) finding similar page-query pairs to reconstruct better queries for search.

first find training web pages similar in content to the target page, then use those pages' click-associated queries to estimate plausible click-associated query content for the page. All the clicked pages in the web query logs and their click-associated queries are used as the training data.

In the SRM based query-side approach, shown in Figure 1.5(b), we add structure to the original unstructured web queries and use an approach similar to that in Figure 1.4(b) to handle the click-through sparseness problem here. Again, the basic idea of this approach is to first discover implicit information for the (now structured) queries and then search with the extended queries. We view both queries and web pages as containing two fields: *Page Content* and *Query Content*. The two shadowed internal boxes in the upper-left external box (query side) in the figure are surrounded by solid lines because we assume that they are observed but incomplete. We also assume the observed *Page Content* and *Query Content* fields contain the same copy of the original query string: intuitively, the query is searching for pages that match the query in

content and/or their click-associated queries. We first use this semi-structured query to find clicked page-query pairs that have similar fields from the training set. After that, we use the information in the similar training pairs to estimate all plausible implicit field values for the query. Finally, we rank all the searched pages by their language modeling based similarity to the extended query.

In Chapter 4, we use the two approaches above to address the click-through sparseness problem, using a publicly available query log sample from Microsoft web search engine to help improve search performance for the standard TREC web search tasks.

1.5 Discovering Implicit Geographic Information in Web Queries

Many times a user's information need has some kind of geographic entity associated with it, or *geographic search intent*. For example, when the user issues the query "coffee amherst", he or she probably wants information about coffee shops only in Amherst, Massachusetts. Using explicit geographic (referred to as "geo" for simplicity) information in the queries can help to personalize web search results, improve a user's search experience and also provide better advertisement matching to the queries. However, research has found that only about 50% of queries with geo search intent have explicit location names (Welch and Cho 2008). Thus, identifying implicit geo intent and accurately discovering missing location information are important for leveraging geo information for search.

Figure 1.6 illustrates our approach for detecting implicit geo intent and predicting city level geo information, using the general perspective from Figure 1.1. Here, the searched items are web pages and the implicit information is city-level geo information. We hypothesize that *implicit geo intent queries may be similar in content to the non-location part of explicit geo intent queries* and that *plausible city level information in the implicit geo intent queries corresponds to the location part of their similar*

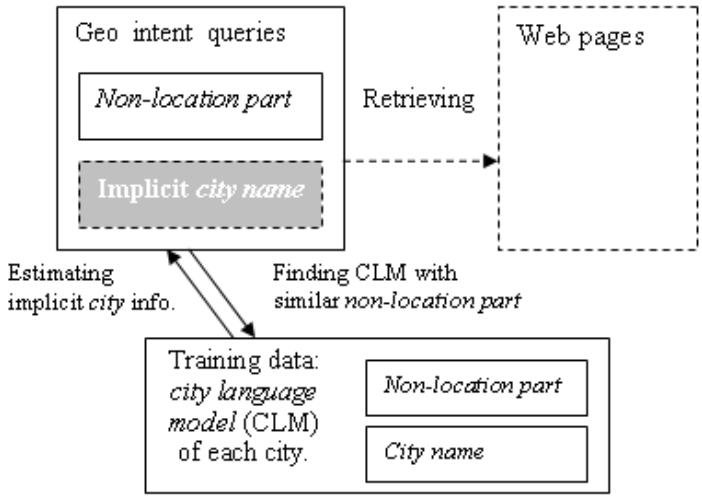


Figure 1.6. The specific perspective of discovering implicit city information in location-specific web queries

explicit geo queries. Under this assumption, we build bi-gram query language models for different cities (called *city language models* or CLMs) from the *non-location* part of explicit geo intent training queries. Then we calculate the posterior of each city language model generating the observed query string (non-location part of the query) for predicting plausible city information and detecting implicit geo search intent.

Previous research has demonstrated how to improve retrieval performance for a query by incorporating related geo information when this information explicitly appears in the query or is known beforehand (Andrade and Silva 2006; Yu and Cai 2007; Jones *et al.* 2008). Therefore, we do not investigate how to incorporate the discovered geo information for retrieval, but explore only finding city-level geo information when it is implicit. Accordingly, we show the retrieval part of the web search task in Figure 1.6 with the dashed line.

In Chapter 5, we use the approach in Figure 1.6 to discover implicit geo information for *simulated* implicit geo intent queries, generated from a large scale industry-level web query log sample from Yahoo! search engine.

1.6 Conclusion

We highlight the specific contributions of our research for each of the four specific IR challenge here. For the implicit field value challenge (introduced in §1.2 and discussed in Chapter 2):

1. We develop a language modeling based technique, called Structured Relevance Models (SRM) that can discover plausible implicit field values in large-scale semi-structured data. We present how to use the discovered information for semi-structured record search task.

2. Using the National Science Digital Library (NSDL) dataset, we empirically show the effectiveness of our technique for discovering implicit field values. In a multi-labeled learning task where the goal is to predict a set of appropriate plausible *subject* values from the whole NSDL collection for synthetic records having empty *subject* fields, our technique correctly predicts 5-6 plausible subject values in its top 8 suggestions and achieves an average precision of 74.5% for selecting the subject field values.

3. We demonstrate the effectiveness of our approach for two real-world semi-structured records search tasks: searching the NSDL collection; and matching semi-structured job and resumes records in an industry-scale online job/resume collection. In the former task, our approach achieves a mean average precision of over 20% when the user-specified query fields are empty in the searched NSDL records; in the latter one, our approach brings more than one matching resume in the top-5 returned results when using job descriptions to search.

4. We design *synthetic* retrieval experiment on the NSDL collection to show that our technique can work with data where the missing field problem is severe in different degrees; the retrieval performance of the technique degrades gradually instead of drastically with more information missing.

For the anchor text challenge (introduced in §1.3 and discussed in Chapter 3):

1. Although content similarity has been used widely in other applications, to the best of our knowledge, we are the first to utilize web content similarity to address the anchor text sparsity problem.

2. We present two complimentary language modeling based techniques – the query-side approach (SRM) and the searched-item-side approach (called relevant anchor text language model or RALM) from our general perspective – to discover plausible anchor text information and use it for retrieval.

3. We design experiments with two large-scale TREC web corpora (GOV2 and ClueWeb09) to demonstrate that RALM can effectively discover missing anchor text for synthetic web pages with no in-links, compared with Metzler et al.'s link-based approach (2009).

4. We use TREC web named-page finding tasks to evaluate the utility of the discovered information by different approaches for helping web search. We show that RALM improves the IR performance by more than 8% over other alternative approaches, including some web hyperlink graph based approaches that discover missing anchor text for a page through collecting anchor text from its web neighbors.

For the click-through challenge (introduced in §1.4 and discussed in Chapter 4):

1. Although content similarity has been used widely in other applications, to the best of our knowledge, we are the first to utilize web content similarity to discover plausible additional semantic click-through features from web query logs for web search.

2. We present two language modeling based approaches based on our general perspective in Figure 1.1 to address the click-through sparseness problem for web

search. The query-side approach is based on SRM and the searched-item-side approach, called relevant (click-associated) query language model (RQLM), is based on using web content similarity. We further combine RQLM and the random walk approach for reducing the click-through sparseness and improving retrieval performance.

3. Using a publicly available query log sample (Microsoft Live Search 2006 Query Log Excerpt) and two sets of TREC ad hoc web search tasks (TREC Terabyte Track 2005-2006 and Web Track 2009-2010), we demonstrate the effectiveness of using our two approaches (SRM and RQLM) to discover additional semantic click-through features from click-through data for web search. For the TREC Web Track ad hoc web search tasks, compared with a standard query likelihood baseline that does not use click-through information, SRM achieves more than 15% improvement of mean average precision (MAP) on the training queries, and RQLM achieves more than 11% improvement of MAP on both the training and testing queries when combined with the state-of-the-art Markov random walk approach (Craswell and Szummer 2007; Gao *et al.* 2009).

4. For this challenge, we show that RQLM is less prone to irrelevant noise in the training web collection than the SRM based approach and achieves better retrieval performance on test queries in both TREC *ad hoc* web search tasks.

For the geo information challenge (introduced in §1.5 and discussed in Chapter 5):

1. We present how to detect web queries' underlying search intent that is implicitly associated with city-level geographic (geo) boundary and discover the corresponding plausible city information, using the query-side approach from our general perspective for handling implicit information. We build geo-related query language models for each city from the non-location part of web queries that explicitly contain the same city, and use the built query language models

(called city language models or CLMs) for our implicit geo search intent analysis task.

2. We generate a large set of *synthetic* implicit city-level geo search intent queries using a large-scale query log sample from the Yahoo! search engine. Then we demonstrate that on these queries, (1) our approach achieves over 90% precision and more than 74% accuracy for detecting implicit geo search intent and (2) the CLMs effectively discovers implicit cities with high precision (88%) and recall (74%). Further human evaluation experiments show CLMs achieves high accuracy (84.5%) of predicting real city labels for the implicit geo intent queries which are highly possibly related to certain particular cities.

Across the tasks and others, our research has the following major general contributions:

1. We present a general perspective for discovering and using implicit data information for different IR challenges. Our approach can be adapted for addressing implicit information for other IR challenges, beyond the four specific challenges investigated in this thesis.

2. We develop language modeling based techniques for effectively discovering implicit information in large-scale real-world textual data for retrieval purposes.

3. We present how to incorporate the discovered information into retrieval process. Using a variety of IR tasks performed on different large-scale real-world datasets, we empirically evaluate the effectiveness of our presented approaches, and demonstrate that using discovered information can improve the retrieval performance for different search tasks.

4. Using our general implicit information discovery perspective, we investigate both the query-side and the searched-item-side approaches of handling implicit

information for several real-world search tasks. The query-side approach introduces a technique called Structured Relevance Models (SRM) to discover implicit information in different query aspects for query expansion. The searched-item-side approach employs a contextual language translation (CLX) approach to discover implicit information in different data aspects of the searched items for smoothing and reranking. We empirically compare their effectiveness and discuss their relative advantages and weaknesses.

1.7 Structure of the Thesis

The remaining parts of the thesis is organized as follows. In the next four chapters, we describe the details of our research on handling the implicit information for the four specific IR challenges, in the same order as we described them in this chapter. Specifically, we present how to discover implicit field values in semi-structured databases for finding relevant records in Chapter 2, how to discover anchor text for web search in Chapter 3, how to discover additional click-through query language information from web query logs for web search in Chapter 4 and how to discover implicit geographic information in web queries in Chapter 5.

Each of these four chapters is organized similarly as follows. First, we provide background of the implicit data information addressed in that chapter and briefly introduce our approach for that task. Then we review related research. Next, we formally describe our language modeling based approach to discover implicit information for each specific IR challenge. When necessary, we also introduce some alternative information discovery approaches to be compared, which use additional information that is only available for that specific search task. After that, we design experiments with large-scale real-world data-sets for evaluating the effectiveness of our proposed approach and comparing it with other available alternative approaches. Finally, we summarize the discovery of our research for that specific IR challenge and conclude.

At the end of this thesis in Chapter 6, we discuss some general observations in our research, such as the advantages and weaknesses of the SRM based query-side and the CLX based searched-item-side approaches of discovering implicit information; then we conclude and discuss some future research directions.

CHAPTER 2

DISCOVERING IMPLICIT FIELD VALUES IN SEMI-STRUCTURED DATABASES FOR SEARCHING RELEVANT RECORDS

2.1 Introduction

In this chapter, we address the challenge of finding relevant records in large-scale semi-structured databases that contain records with incomplete or empty fields. We start with a detailed description of this research issue.

Information processing of *semi-structured* data is a prominent research area in both the IR and Relational Databases (DB) research fields. "Semi-structured" typically means that the data have some semantic structures, e.g. tags and markups, but do not conform with the formal structure of tables and data models associated with typical database systems (Buneman 1997). For example, HTML/XML documents and emails are some types of semi-structured data because they use simple HTML/XML tags or email fields, respectively, to denote semantic units in the documents instead of formal relational structure of tables and data models. Here we consider semi-structured documents/records that contain natural language textual fields. For example, if documents contained *subject* and *author* fields, we might see queries looking for documents about the *theory of relativity* by the author *Einstein*.

Some relational database research combined the Structured Query Language (SQL) and some typical IR relevance metrics (e.g. $tf \cdot idf$ score) to use a structured Boolean relational query for searching and ranking semi-structured documents (e.g., Grabs and Schek (2002)). For example, a structured query like: *subject = 'elementary differential geometry' AND audience = 'undergraduate'* might be formulated to answer

a user's request for finding undergraduate reading materials about elementary differential geometry. Then the returned documents are ranked according to the query terms' $tf \cdot idf$ scores in each document's field (Grabs and Schek 2002). However, such an Boolean field matching SQL-like approach usually assumes complete information for every record in the database, while in many real-world semi-structured documents, many text fields are *incomplete* or even *missing*. Thus, this approach might miss many plausible relevant documents about *'elementary differential geometry'*, only because they lack the target *audience* (reading level) information. Our research aims to find all plausible relevant information in response to a query such as the one above.

Our research is motivated by the challenges we encountered in working with the National Science Digital Library (NSDL) collection.[1] Each item in the collection is a scientific resource, such as a research paper or an educational video. In addition to its main content, each resource is annotated with *metadata*, which provides information such as the author or creator of the resource, its subject area, format (text/image/video) and intended audience – in all over 90 distinct fields. Making use of such extensive metadata in a digital library paves the way for constructing highly-focused models of the user's information need. These models have the potential to dramatically improve the user experience in targeted applications, such as the NSDL portals.

However, directly using a relational engine for searching a semi-structured collection similar to the NSDL collection will run into a number of obstacles. One problem is that natural language fields are filled inconsistently: e.g., the *audience* field may contain values such as *K-4*, *K-6*, *second grade*, and *learner*, all of which are clearly semantically related. A larger problem is that of empty fields. Table 2.1 shows some

[1] http://www.nsdl.org

	records covered	average length	unique words
title	655,673 (99%)	7	102,772
description	514,092 (78%)	38	189,136
subject	504,054 (77%)	12	37,385
content	91,779 (14%)	743	575,958
audience	22,963 (3.5%)	4	119

Table 2.1. Summary statistics for the five NSDL fields used in our experiments.

statistics of 5 fields (title, description, subject, content and audience) from a January 2004 snapshot of the NSDL collection. It can be observed that 23% of the records in the collection have *empty* subject field and only 3.5% mention target audience. Therefore if a relational engine were directly applied for querying records in the NSDL collection, it will bump into the empty field problem. For example if a query contains *audience = 'elementary school'*, it will consider at most 3.5% of all potentially relevant resources in the NSDL collection.

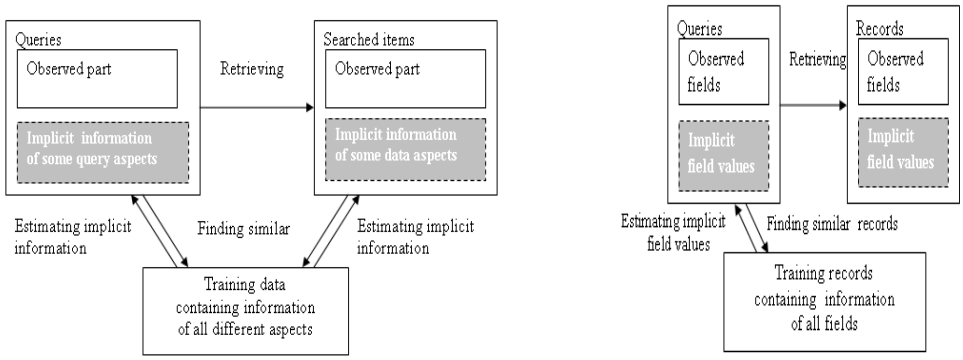


Figure 2.1. Discovering implicit field values for semi-structured records following the general perspective for discovering implicit information

To address the above issue, following our general perspective from Figure 2.1(a)(also shown in Figure 1.1 in Chapter 1), we employ the approach of Figure 2.1(b) to discover implicit field values for the semi-structured record retrieval tasks. Our hypothesis is that *semi-structured records that have similar attribute values in some fields may have*

similar attribute values in other fields due to the cross-field relations between attribute values in different fields. Using this assumption, we discover plausible implicit field values of semi-structured records by using their similar records' corresponding information. Then the inferred information can be used for retrieval. We develop language modeling based technique to estimate the likely implicit field values for every empty field in a given query, based on the context of the observed query fields.

We evaluate the performance of our approach by investigating two different retrieval tasks on two large-scale real-world semi-structured databases that contain incomplete data records. The first is the IR challenge on the National Science Digital Library (NSDL) collection described at the beginning of this chapter. The second is to match semi-structured job and resume records in a large scale job/resume collection provided by Monster Worldwide[2].

The remaining parts of this chapter will be organized as follows. We begin by reviewing related work in §2.2. In §2.3, we formally describe a *hypothetical* probabilistic procedure of generating semi-structured records and present how to use this procedure to estimate the distributions of plausible implicit field values in semi-structured data. Next, in §2.4 we design synthetic experiments using the NSDL collection mentioned earlier to directly evaluate the quality of the discovered field values by our approach; for comparison, we also report the evaluation results of using an alternative machine learning approach on the simulated data. These results demonstrate the potential of using our approach for retrieval. After that, in §2.5 we describe the details of how to employ our technique for retrieval. In §2.5.2, we evaluate the retrieval performance of our approach on the NSDL search task. In §2.5.3, we design a small-scale synthetic IR experiment with the NSDL records to evaluate how our approach performs when encountering different amount of missing information in the semi-structured data. After

[2] `http://www.monster.com`, an online job service company

that, in §2.5.4 we employ our approach for another large-scale semi-structured data search task: matching suitable job/resume pairs in the Monster data. We conclude in §2.6.

2.2 Related Work

The issue of handling missing field values in semi-structured data is addressed in a number of publications straddling the areas of relational databases and machine learning. Researchers usually introduce a statistical model for predicting the value of a missing attribute or relation, based on observed values. Friedman et al. (1999) introduced a directed graphical model, Probabilistic Relational Models (PRM) that extends Bayesian networks for automatically learning the structure of dependencies and reasoning in a relational database. Taskar et al. (2001) demonstrated how PRM can be used to predict the category of a given research paper and show that categorization accuracy can be substantially improved by leveraging the relational structure of the data. They also proposed a technique called relational Markov networks (RMNs) (Taskar *et al.* 2002), which use undirected graphical models for reasoning with autocorrelation in relational data. Heckerman et al. (2004) introduced the Probabilistic Entity Relationship model as an extension of PRM that treats relations between entities as objects. Neville et al. proposed several relational learning models, including Relational Bayesian Classifier (RBC) (Neville *et al.* 2003), Relational Probabilistic Trees (RPT) (Neville *et al.* 2003) and Relational Dependency Networks (RDN) (Neville and Jensen 2003), to predict unknown (or missing) attribute values of some records in relation databases, based on different assumptions of the dependencies in relational data. Different from these approaches, we work with free-text fields that contain thousands of different field values (words), whereas relational learning tasks usually deal with closed-vocabulary values, which usually exhibit neither the synonymy nor the polysemy inherent in natural language expressions.

Discovering multiple implicit field values can be viewed as a multi-labeled classification problem in machine learning (ML) research where each unique field value represents a different label and each record has multiple labels. The challenging goal is to automatically classify each data sample into more than one category. Zhu et al. (2005) provided a detailed survey for different approaches of multi-labeled classification techniques. Some research built complicated hierarchical discriminative learning models (Godbole and Sarawagi 2004; Rousu *et al.* 2006) while our research follows a generative approach for this classification problem. The generative approach typically relies on some hypothetical generative probabilistic model to generate samples, and learns posteriors for classification. McCallum (1999) described a parametric generative mixture model which assumes that each multi-labeled sample is generated by a mixture of single-labeled generative models, then utilized EM algorithm for learning parameters. Different from this research, we focus on the specific task of discovering implicit values in semi-structured database and develop our technique based on a probabilistic procedure of generating semi-structured records. We also directly handle large scale incomplete semi-structured data where there are a large number of empty fields. Furthermore, the goal of our work is different: we aim for using discovered field values for retrieval purpose, i.e., accurately *ranking* incomplete records by their relevance to the user's query. Our approach is related to the relevance based language models (RMs), proposed by Lavrenko and Croft (2001). Their original work introduces the RMs to discover plausibly useful query terms for query expansion while our approach further leverages the structure in the queries and searched records for building structured RMs and searching relevant records.

Our work is also related to a number of existing approaches for semi-structured text search. Desai et al. (1987) followed by Macleod (1991) proposed using the standard relational approach to searching semi-structured texts. The lack of an explicit ranking function in their approaches was partially addressed by Blair (1988). Fuhr

(1993) proposed the use of Probabilistic Relational Algebra (PRA) over the weights of individual term matches. Vasanthukumar et al. (1996) developed a relational implementation of the inference network retrieval model. A similar approach was taken by de Vries and Wilschut (1999), who managed to improve the efficiency of the approach. De Fazio et al. (1995) integrated IR and RDBMS technology using an approached called cooperative indexing. Cohen (2000) described WHIRL – a language that allows efficient inexact matching of textual fields within SQL statements. A number of relevant works have been published in the proceedings of the *INEX* workshop.[3] The main difference between these endeavors and our work is that we are explicitly focusing on the cases where parts of the structured data are incomplete or missing. For the situation where the original queries do not have explicit field structure, Kim et al. (2009) proposed a language modeling based approach to discover the implicit query field structure for better searching relevant records. Their research complements our work which focuses on the implicit field values in queries and searched records.

Our approach for discovering plausible implicit field values for retrieval was initially presented in one published paper (Lavrenko *et al.* 2007), which focused on searching relevant semi-structured NSDL records where both the query and its relevant records may contain incomplete or empty fields. In this paper, Lavrenko designed a hypothetical process of generating semi-structured records in the language modeling framework and proposed a retrieval technique based on this generative process; then we together implemented the retrieval technique and designed retrieval experiments with the NSDL collection to evaluate the performance of the technique. The experimental results are also described in §2.5.2 in this chapter. I did further experiments to evaluate the robustness of our approach when encountering different amount of missing information, and the results are presented in §2.5.3 in this chapter.

[3] `http://inex.is.informatik.uni-duisburg.de/index.html` and `http://www.informatik.uni-trier.de/~ley/db/conf/inex/`

We further investigated directly discovering implicit field values in semi-structured databases using our approach and compared its performance with several state-of-the-art-relational learning approaches (Yi *et al.* 2007). Part of our results are described in §2.4 in this chapter. Moreover, we applied this technique for the IR challenge of matching appropriate resume/job pairs in the semi-structured Monster dataset that also contain large amounts of incomplete or empty fields (Yi *et al.* 2007). This work is also described in §2.5.4 in this chapter.

2.3 Discovering Implicit Field Values

In this section we provide a detailed description of our generative approach to address the existing empty field problem when searching semi-structured records. The search task here is to identify a set of records relevant to a semi-structured query provided by the user. We assume the query specifies a set of keywords for each field of interest to the user, for example *Q: subject='physics,gravity' AND audience='grades 1-4'*[4] Each record in the database is a set of natural-language descriptions for each field. A record is considered relevant if it *could plausibly* be annotated with the query fields. For example, a record clearly aimed at elementary school students would be considered relevant to Q even if it does not contain *'grades 1-4'* in its description of the target audience.

This task is not a typical search task because the fielded structure of the query is a critical aspect of the processing, not one that is largely ignored in favor of pure content based retrieval. On the other hand, the approach used is different from most DB work because we explicitly target the empty field problem.

Our approach is based on the idea that plausible values for a given field could be inferred from the context provided by the other fields in the record. For instance,

[4]Here we will focus on simple conjunctive queries. Extending our model to more complex queries is reserved for future research.

a resource titled *'Transductive SVMs'* and containing highly technical language in its description is unlikely to be aimed at elementary-school students. Next in §2.3.1 and §2.3.2 we will describe a statistical model that will allow us to infer the values of un-observed fields. At the intuitive level, the model takes advantage of the fact that records similar in one respect will often be similar in others. For example, if two resources share the same author and have similar titles, they are likely to be aimed at the same audience. Formally, our model is based on the *generative* paradigm where we assume a probabilistic process that could be viewed, hypothetically, as the source of every record in our collection.

2.3.1 Definitions

We start with a set of definitions that will be used through the remainder of this chapter. Let $\mathcal{C}$ be a collection of semi-structured records. Each record $\mathbf{w}$ consists of a set of fields $\mathbf{w}_1 \ldots \mathbf{w}_m$. Each field $\mathbf{w}_i$ is a sequence of discrete variables (words) $\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,n_i}$, taking values in the field vocabulary $\mathcal{V}_i$.[5] When a record contains no information for the i'th field, we assume $n_i{=}0$ for that record. We will use $\mathbf{p}_i$ to denote a *language model* over $\mathcal{V}_i$, i.e. a set of probabilities $\mathbf{p}_i(v){\in}[0,1]$, one for each word v, obeying the constraint $\Sigma_v \mathbf{p}_i(v) = 1$. The set of all possible language models over $\mathcal{V}_i$ will be denoted as the probability simplex $I\!P_i$. We define $\pi : I\!P_1 {\times} \cdots {\times} I\!P_m {\rightarrow} [0,1]$ to be a discrete measure function that assigns a probability mass $\pi(\mathbf{p}_1 \ldots \mathbf{p}_m)$ to a set of m language models, one for each of the m fields present in our collection.

2.3.2 Generative Model

We now present a generative process that will be viewed as a hypothetical source that produced every record in the collection $\mathcal{C}$. We stress that this process is purely *hypothetical*; its only purpose is to model the kinds of dependencies that are useful

[5] We allow each field to have its own vocabulary $\mathcal{V}_i$, since we generally do not expect author names to occur in the audience field, etc. We also allow $\mathcal{V}_i$ to share words.

for inferring implicit field values from observed parts of a record. We assume that each record $\mathbf{w}$ in the database is generated in the following manner:

1. Pick m distributions $\mathbf{p}_1 \ldots \mathbf{p}_m$ according to π
2. For each field $i = 1 \ldots m$:
   (a) Pick the length n_i of the $i'th$ field of $\mathbf{w}$
   (b) Draw i.i.d. words $\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,n_i}$ from $\mathbf{p}_i$

Under this process, the probability of observing a record $\{\mathbf{w}_{i,j} : i{=}1..m, j{=}1..n_i\}$ is given by the following expression:

$$P(\mathbf{w}) = \int_{I\!P_1 \ldots I\!P_m} \left[\prod_{i=1}^{m} \prod_{j=1}^{n_i} \mathbf{p}_i(\mathbf{w}_{i,j}) \right] \pi(\mathbf{p}_1 \ldots \mathbf{p}_m) d\mathbf{p}_1 \ldots d\mathbf{p}_m \tag{2.1}$$

2.3.2.1 A Generative Measure Function

The generative measure function π plays a critical part in Equation (2.1): it specifies the likelihood of using different combinations of language models in the process of generating $\mathbf{w}$. The measure function can be set in a number of different ways, leading to very different dependence structures among the fields of $\mathbf{w}$. In choosing π we tried to make as few assumptions as possible about the structure of our collection, allowing the data to speak for itself. We use a non-parametric estimate for π, which makes our generative model similar to *Parzen windows* or *kernel-based* density estimators (Silverman 1986).[6] Our estimate relies directly on the combinations of language models that are observed in the training part of the

[6]The distinguishing feature of our model is that it operates over discrete events (strings of words), and accordingly the mass function is defined over the space of language models, rather than directly over the data points, as would be done by a Parzen window.

collection. Each training record $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_m$ corresponds to a unique combination of language models $\mathbf{p}_1^{\mathbf{w}} \ldots \mathbf{p}_m^{\mathbf{w}}$ defined by the following equation:

$$\mathbf{p}_i^{\mathbf{w}}(v) = \frac{\#(v, \mathbf{w}_i) + \mu_i c_v}{n_i + \mu_i} \tag{2.2}$$

Here $\#(v, \mathbf{w}_i)$ represents the number of times the word v was observed in the i'th field of $\mathbf{w}$, n_i is the length of the i'th field, and c_v is the relative frequency of v in the entire collection. Dirichlet smoothing parameters μ_i (Zhai and Lafferty 2001b) allow us to control the amount of smoothing applied to language models of different fields; their values are set empirically on a held-out portion of the data.

We define $\pi(\mathbf{p}_1 \ldots \mathbf{p}_m)$ to have mass $\frac{1}{N}$ when its argument $\mathbf{p}_1 \ldots \mathbf{p}_m$ corresponds to one of the N records $\mathbf{w}$ in the training part $\mathcal{C}_{tn}$ of our collection, and zero otherwise:

$$\pi(\mathbf{p}_1 \ldots \mathbf{p}_m) = \frac{1}{N} \sum_{\mathbf{w} \in \mathcal{C}_{tn}} \prod_{i=1}^{m} 1_{\mathbf{p}_i = \mathbf{p}_i^{\mathbf{w}}} \tag{2.3}$$

Here $\mathbf{p}_i^{\mathbf{w}}$ is the language model associated with the training record $\mathbf{w}$ (equation 2.2), and 1_x is the δ Boolean indicator function that returns 1 when its predicate x is true and zero when it is false. Note that by using this generative measure function π, the integral in Equation (2.1) is not the *Riemann* integral but the *Lebesgue* integral and the probability of observing a new record $\{\mathbf{w}' = \mathbf{w}'_{i,j} : i{=}1..m, j{=}1..n_i\}$ is:

$$P(\mathbf{w}') = \frac{1}{N} \sum_{\mathbf{w} \in \mathcal{C}_{tn}} \prod_{i=1}^{m} \prod_{j=1}^{n_i} \mathbf{p}_i^{\mathbf{w}}(\mathbf{w}'_{i,j}) \tag{2.4}$$

2.3.2.2 Assumptions and Limitations of the Model

The generative model described in the previous section treats each field in the record as a *bag* of words with no particular order. This representation is often associated with the assumption of *word independence*. We would like to stress that our model does not assume word independence, on the contrary, it allows for strong

un-ordered dependencies among the words – both within a field, and across different fields within a record. To illustrate this point, suppose we let $\mu_i \to 0$ in Equation (2.2) to reduce the effects of smoothing. Now consider the probability of observing the word *'elementary'* in the audience field together with the word *'differential'* in the title (Equation 2.4). It is easy to verify that the probability will be non-zero only if some training record $\mathbf{w}$ actually contained these words in their respective fields – an unlikely event. On the other hand, the probability of *'elementary'* and *'differential'* co-occurring in the same title might be considerably higher.

While our model does not assume word independence, it does ignore the relative ordering of the words in each field. Consequently, the model will fail whenever the order of words, or their proximity within a field carries a semantic meaning.

2.3.3 Estimating Plausible Implicit Field Values

Now we utilize the generative model described above to estimate the distributions over plausible values $v \in \mathcal{V}_i$ in different fields $\mathbf{w}_i$ of a semi-structured record $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_m$. Assume that the whole collection $\mathcal{C}$ has been divided into the training part $\mathcal{C}_{tn}$ and the testing part $\mathcal{C}_{tt}$. Given a testing record $\mathbf{w}' \in \mathcal{C}_{tt}$, we now use the training part $\mathcal{C}_{tn}$ to estimate plausible implicit field values for $\mathbf{w}'$ by using the observed $\mathbf{w}'_1 \ldots \mathbf{w}'_m$. Specifically, we calculate a set of *relevance models* $R_1 \ldots R_m$ for $\mathbf{w}'$, where the relevance model $R_i(v)$ specifies how plausible it is that word v would occur in the i'th field of $\mathbf{w}'$ given the observed $\mathbf{w}' = \mathbf{w}'_1 \ldots \mathbf{w}'_m$, by:

$$R_i(v) = P(\mathbf{w}'_1 \ldots v \circ \mathbf{w}'_i \ldots \mathbf{w}'_m) / P(\mathbf{w}'_1 \ldots \mathbf{w}'_i \ldots \mathbf{w}'_m). \tag{2.5}$$

We use $v \circ \mathbf{w}'_i$ to denote appending word v to the string $\mathbf{w}'_i$. We call the estimated distributions $\mathbf{R}(\mathbf{w}') = R_1 \ldots R_m$ *Structured Relevance Models* (**SRM**) for $\mathbf{w}'$, since they may provide all plausible relevant information that can be inferred from the observed parts of the record $\mathbf{w}'$.

To calculate the SRM $\mathbf{R}(\mathbf{w}')$ for $\mathbf{w}'$, we can rewrite Equation (2.5) as:

$$R_i(v) = (\sum_{\mathbf{w}\in\mathcal{C}_{tn}} P(\mathbf{w}'_1 \ldots v \circ \mathbf{w}'_i \ldots \mathbf{w}'_m|\mathbf{w}) * P(\mathbf{w}))/P(\mathbf{w}'). \tag{2.6}$$

According to the generative process and Equation (2.4), we further have:

$$\begin{gathered} R_i(v) = \textstyle\sum_{\mathbf{w}\in\mathcal{C}_{tn}} \mathbf{p}_i^{\mathbf{w}}(v) * P(\mathbf{w}|\mathbf{w}'), \\ P(\mathbf{w}|\mathbf{w}') \propto P(\mathbf{w}'|\mathbf{w}). \end{gathered} \tag{2.7}$$

In this way, the SRM $\mathbf{R}(\mathbf{w}')$ can be computed using Equation (2.2) and (2.7). Similar to how the typical relevance models are implemented in practice (Lavrenko and Croft 2001), for efficiency $\mathbf{R}(\mathbf{w}')$ is computed from the top-k most similar records of $\mathbf{w}'$, i.e. records that have the top-k highest posterior probabilities $P(\mathbf{w}|\mathbf{w}'_1 \ldots \mathbf{w}'_i \ldots \mathbf{w}'_m)$ in Equation (2.7). We can use this approximation method because that the posteriors of other records in Equation (2.7) are relatively small thus have little impact on the result $R_i(v)$. We tune the value of k on training data in different experiments.

2.4 Evaluating Discovered Field Values

In this section, we focus on directly evaluating the quality of the discovered plausible field values by using the Structured Relevance Models approach. The purpose here is to investigate the *potential* of using our approach for retrieval. Employing the proposed technique for real-world retrieval tasks will be described later in §2.5.

We can use the value $R_i(v)$ of each plausible value v, i.e. v's estimated occurrence probability in the i'th field of a semi-structured record $\mathbf{w}$, in the computed SRM $\mathbf{R}(\mathbf{w})$ to rank the relative importance of v in the i'th field of $\mathbf{w}$. Thus through analyzing the quality of the top-k most important values in each field according to the SRM, we can evaluate its effectiveness of discovering plausible implicit field values for semi-structured records. From the view of machine learning research, predicting multiple

un-observed field values is a multi-labeled classification problem where each unique field value represents a different category label and each record has multiple labels. Our SRM approach follows a *generative* approach for this classification problem where we use a hypothetical generative model for estimating the probability of each record belonging to each category. Alternatively, we could have followed a *discriminative* approach in the machine learning research for predicting multiple un-observed labels (Tang *et al.* 2009), where we can build discriminative classifiers for each category (known as One-Vs-Rest approach) and use the trained classifiers for the prediction. However, it will induce high computational cost to employ the discriminative approach to predict plausible textual field values: these fields usually consist of free text instead of closed-vocabulary small-sized labels. For example, we can see in Table 2.1 (in §2.1) that there are 119 categories in the *audience* field and 37,385 categories in the *subject* field, from the view of the multi-labeled learning. Nevertheless, we use a *small-scale subset* records from the NSDL snapshot to compare the performance of the SRM approach and the discriminative learning approach. Only a limited number of NSDL records are selected for this comparison experiment because training and testing with the whole NSDL collection (656,992 records) with huge label variety are prohibitively expensive using the discriminative approach. We then move to a large-scale experiment, where the whole NSDL collection is used and only the SRM approach can be employed.

In the small-scale experiment, we confine ourselves to a subset of the NSDL collection. This subset includes all the NSDL records in which all five fields (*title*, *content*, *description*, *subject* and *audience*) in Table 2.1 are *non-empty* – overall there are 11,596 of these records. The multi-labeled learning task is to predict the *subject* field values of each NSDL record given its *title*, *content* and *description* fields' information, i.e. we assume the subject values are *missing* for these records and the original subject values are then used as ground-truth values for evaluation. Because

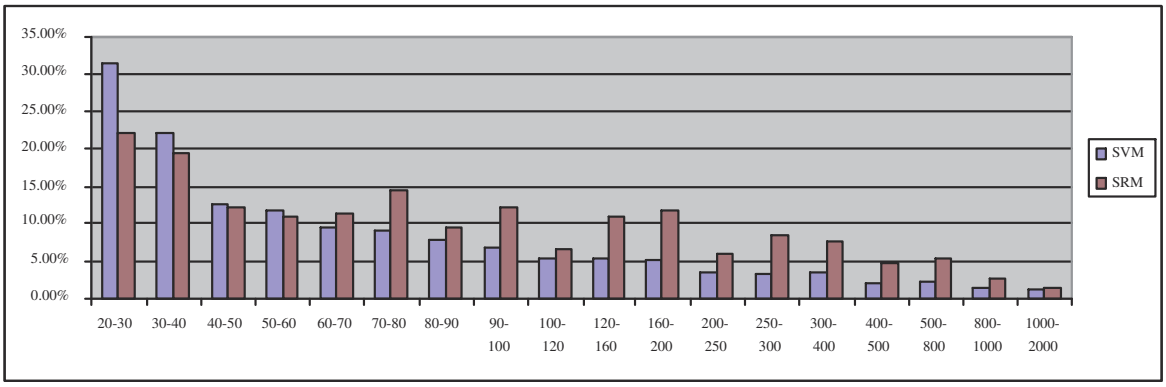


Figure 2.2. Average *error* rates for the SRM and SVM approaches to selecting the *subject* field values, as a function of the number of records of a subject label there are in the corpus.

of the computational cost issue of the discriminative approach, we only consider the 211 most frequent values in the subject field of the whole NSDL collection – each subject value is viewed as an individual category. The discriminative approach we employ for this task is the multi-labeled Support Vector Machines (SVM) (Chang and Lin 2006), which is similar to the approach used by Tang et al. (2009) for query classification.

We use 5-fold cross validation and calculate the per-value average error rates for both the SVM approach and the SRM approach, as a function of the number of records of the subject value there are. Errors are measured as the proportion of *incorrect* labels that are ranked higher than the one being measured. Figure 2.2 and Table 2.2 shows, for example, that if a subject value occurs 20-30 times, the SVM error rate is 31% compared to only 22% for our approach; on the other hand, with 160-200 records, the error rates are 5% and 12%, respectively. The results show that SRM can achieve lower error rates than the SVM approach on values that appear less frequently in the records.

Table 2.3 further shows the statistics for the number of records of different *subject* values in the original NSDL collection. We can observe that more than 90% of the

Num of records		20-30	30-40	40-50	50-60	60-70	70-80
SVM	avg	31.37%	22.15%	12.57%	11.66%	9.49%	9.13%
	stdev	2.58%	2.23%	2.24%	2.16%	2.37%	3.33%
SRM	avg	22.19%	19.51%	11.91%	10.80%	11.19%	14.55%
	stdev	2.25%	2.71%	1.74%	1.78%	3.51%	1.67%
Num of records		80-90	90-100	100-120	120-160	160-200	200-250
SVM	avg	7.90%	6.65%	5.25%	5.40%	5.04%	3.46%
	stdev	4.75%	3.22%	0.87%	2.52%	1.54%	1.46%
SRM	avg	9.50%	12.04%	6.46%	10.72%	11.68%	5.96%
	stdev	1.06%	1.79%	0.86%	1.70%	1.35%	1.37%
Num of records		250-300	300-400	400-500	500-800	800-1000	1000-2000
SVM	avg	3.15%	3.29%	1.94%	2.09%	1.37%	1.12%
	stdev	1.03%	0.93%	0.96%	0.86%	1.12%	0.38%
SRM	avg	8.37%	7.48%	4.70%	5.32%	2.60%	1.39%
	stdev	1.05%	1.50%	0.24%	0.21%	0.47%	0.06%

Table 2.2. Averages and standard deviations of the *error* rates for the SRM and SVM approaches to selecting the *subject* field values.

Bin	0-10	10-20	20-30	30-40	40-50	50-60	60-70
Frequency	28724	2476	1110	658	476	355	264
Cumulative%	76.83%	83.46%	86.43%	88.19%	89.46%	90.41%	91.11%
Bin	70-80	80-90	90-100	100-120	120-160	160-200	200-250
Frequency	210	176	187	268	348	270	203
Cumulative%	91.68%	92.15%	92.65%	93.36%	94.29%	95.02%	95.56%
Bin	250-300	300-400	400-500	500-800	800-1000	1000-2000	>2000
Frequency	158	249	172	309	114	248	410
Cumulative%	95.98%	96.65%	97.11%	97.94%	98.24%	98.90%	100.00%

Table 2.3. Statistics for the number of records of different subject values.

subject values have fewer than 60 instances. This indicates that for the free-text field value discovery in the large-scale collection, SRM can be a more appropriate choice than SVM for the major portion of *unique* field values, in terms of both effectiveness and computational cost.

Now we consider an environment where the SVM training approach was prohibitively expensive in terms of time. For each test record, we ask the system to predict a subset of *subject* field values from the 37,385 possible values in the NSDL collection for a semi-structured record. That number of possible values is sufficiently

Examples		Term frequency	Ground-truth terms	$R_{subject}(v)$	Ordered term lists
Subject	1	3	wave	0.145	wave*
		2	interference	0.132	physics*
		1	physics	0.053	probable*
		1	quantum	0.051	interference*
		1	tutorial	0.039	quantum*
		1	particle	0.037	tutorial*
		1	slit	0.028	travel
		1	probable	0.022	slit*
	2	1	astronomy	0.348	astronomy*
		1	historic	0.175	general
		1	myth	0.174	historic*
		1	legend	0.173	constellate*
		1	constellate	0.006	physics
	3	1	science	0.182	calculus*
		1	theory	0.137	variable
		1	precalculus	0.137	single
		1	calculus	0.0167	science*
		1	linear	0.0164	multivariable
		1	algebra	0.0162	geometry
		1	number	0.0131	compute

Table 2.4. Some examples of employing SRM for discovering plausible *subject* field values. For each record, Column 3 shows the *true* field values of that record, Column 5 and Column 4 show top-N term lists returned by SRM (cut by the number of *true* field values for subject field) and their corresponding probabilities in SRM, respectively. * indicates the predicted subject value is correct according to Column 3.

large that we could not reasonably train an SVM for each one as we did in the small-scale experiment. We just report results for the SRM approach.

To evaluate this, we randomly select 1,122 records (10% of our earlier set) that had all five fields. For each record, we again use the *content*, *description*, and *title* fields to predict its plausible *subject* values. We use the remaining 655,870 records as training data for building SRM, though it is important to note: 23% of training records have no *subject* field (i.e. category label information is missing in them), and some feature information used for prediction are also missing in the training data, e.g. 22% of the records are missing *description* field and 86% lack a *content* field. Each test record can have multiple values of the field (on average, records contain 12 *subject* values). The system's output is a ranked list of subject values. Table 2.4 presents some example outputs of the predicted plausible values by SRM in this experiment.

To evaluate the system's output, we use standard IR measures, including Precision at k (P@k), Recall-Precision (R-Precision), based on where in the ranked list the *correct* values occur. "P@k" measures the proportion of correct subject values listed in the top k items. The "R-precision" value measures the proportion of correct suggestions in the top R listed, where R is the actual number of suggestions that would have ideally been found. The experimental results are shown in Table 2.5. P@5$\approx$ 46% shows that almost half of the subject values listed in the top five items suggested are correct. R-precision $\approx$ 70% shows that about 70% suggestions in the top R listed are correct, e.g. if a record has 8 subjects assigned (in the truth), then on average the top 8 suggestions would have included 5-6 (70%) that are correct. These results indicate that the SRM approach is very effective at discovering plausible implicit values in free-text fields in large-scale semi-structured data, thus has very promising potential to be used for retrieval.

P@5	0.4617
P@20	0.1496
R-precision	0.7010

Table 2.5. Using SRM for discovering missing subject values in the NSDL collection.

2.5 Searching Incomplete Semi-structured Databases

In this section, we first formally describe how we use SRM for search tasks on semi-structured records that may contain incomplete fields, following our approach depicted in Figure 2.1 in the introduction of this chapter (also shown in Figure 1.3 in Chapter 1). Then we apply our approach for different retrieval tasks, including both synthetic and real-world ones, and evaluate its retrieval performance when encountering different amounts of missing field values.

2.5.1 Retrieval Procedure

Our approach is to discover plausible implicit field values in the semi-structured queries given their observed field information (as shown in Figure 2.1(b)), and then search relevant semi-structured records (complete or incomplete) in the database by matching them to the extended queries. Here we formally describe this approach.

Suppose that a user's query $\mathbf{q}$ takes the same representation as a semi-structured record $\mathbf{w}$ (described in §2.3.1) in the collection $\mathcal{C}$, i.e., $\mathbf{q}=\{\mathbf{q}_{i,j}\in\mathcal{V}_i : i=1..m, j = 1..n_i\}$ and that $\mathcal{C}$ has been partitioned into the training portion $\mathcal{C}_{tn}$ and the testing portion $\mathcal{C}_{tt}$. Now we assume that the input query is also a sample from the semi-structured record generative process described in §2.3.2, albeit a very short one. Using this assumption, we can employ the probabilistic model in §2.3.2 to discover plausible implicit words in each query field. Then we can reconstruct a query with the discovered information for better searching relevant records.

Specifically, we use the same technique described in §2.3.3 to calculate a Structured Relevance Model: $\mathbf{R}(\mathbf{q})= R_1 \ldots R_m$ for an input query $\mathbf{q} = \mathbf{q}_1 \ldots \mathbf{q}_m$ to better reflect the user's information need. Each relevance model $R_i(v)$ specifies how plausible it is

that word v would occur in the i'th field $\mathbf{q}_i$ of a record, given that the record contains a perfect match to the query fields $\mathbf{q}_1 \ldots \mathbf{q}_m$:

$$R_i(v) = P(\mathbf{q}_1 \ldots v \circ \mathbf{q}_i \ldots \mathbf{q}_m | \mathbf{q}_1 \ldots \mathbf{q}_i \ldots \mathbf{q}_m), \tag{2.8}$$

where $v \circ \mathbf{q}_i$ denotes appending word v to the string $\mathbf{q}_i$. $R_i(v)$ in Equation (2.8) is computed using Equation (2.6) and (2.7) (in §2.3.3). At the intuitive level, the SRM $\mathbf{R}(\mathbf{q})$ is computed using the field information from $\mathbf{q}$'s similar records that have highest posteriors $P(\mathbf{w}|\mathbf{q}_1 \ldots \mathbf{q}_i \ldots \mathbf{q}_m)$, as shown in Figure 2.1(b). In practice, as we discussed in §2.3.3, we only use the top-k most similar records of $\mathbf{q}$ to compute the $R_i(v)$ for efficiency. The value of k is tuned on a held-out portion of the data.

After we compute the SRM $\mathbf{R}(\mathbf{q})$, we can rank testing records $\mathbf{w}' \in \mathcal{C}_{tt}$ by their similarity to it. As a similarity measure we use weighted cross-entropy, which is an extension of the ranking formula originally proposed by Lafferty and Zhai (2001):

$$H(R_{1..m}; \mathbf{w}'_{1..m}) = \sum_{i=1}^{m} \alpha_i \sum_{v \in \mathcal{V}_i} R_i(v) \log \mathbf{p}^{\mathbf{w}'_i}(v). \tag{2.9}$$

The outer summation goes over every field of interest, while the inner extends over all the words in the vocabulary of the i'th field. $\mathbf{p}^{\mathbf{w}'_i}$ are estimated from Equation (2.2). Meta-parameters α_i allow us to vary the importance of different fields in the final ranking; the values are also tuned on a held-out portion of the data.

2.5.2 Retrieval Experiment on the NSDL Snapshot

2.5.2.1 Data and Methodology

We first employ our SRM-based retrieval approach for a search task on the NSDL collection (described in §2.1). We randomly split the NSDL snapshot into three subset collections for the retrieval experiments: the **training** set, which contains 50% of the records and is used for building the SRM; the **held-out** set, which comprises 25%

of the data and is used to tune the smoothing parameters μ_i and the bandwidth parameters α_i for the i'th field of records; and the **testing** set, which contains 25% of the records and is used to evaluate the performance of the tuned model[7].

Our experiments are based on a set of 127 semi-structured queries. The queries were constructed by combining some randomly picked *subject* words with some *audience* words, and then discarding any combination that had less than 10 *exact* matches in any of the three subsets of our collection. This procedure yields queries such as Q_{91}={*subject='artificial intelligence' AND audience='researchers'*}, or Q_{101}= {*subject='philosophy' AND audience='high school'*}. Then we randomly split the queries into two groups, 64 for training and 63 for evaluation.

We evaluate SRM's ability to find "relevant" records in the face of empty fields. In this experiment, we define a record **w** to be relevant to the user's query **q** if every keyword in **q** is found in the corresponding field of **w**. For example, in order to be relevant to Q_{101} a record must contain the word *'philosophy'* in the subject field and words *'high'* and *'school'* in the audience field. If either of the keywords is missing, the record is considered non-relevant.[8]

When the *subject* and *audience* fields of testing records are fully observable, achieving perfect retrieval accuracy is trivial: we simply return all records in the testing set that match all query keywords in the subject and audience fields. However, our main interest concerns the scenario when parts of the testing data are missing. We are going to *simulate* this scenario in a rather extreme manner by *completely* removing the *subject* and *audience* fields from all testing records. This means that a straight-

[7]In real use, a typical pseudo relevance feedback scheme can be followed: retrieve top-k documents to build the SRM then perform IR again on the same whole collection.

[8]This definition of relevance is unduly conservative by the standards of IR researchers. Many records that might be considered relevant by a human annotator will be treated as non-relevant, artificially decreasing the accuracy of any retrieval algorithm. However, our approach has the advantage of being fully automatic: it allows us to test different models on a scale that would be prohibitively expensive with manual relevance judgments.

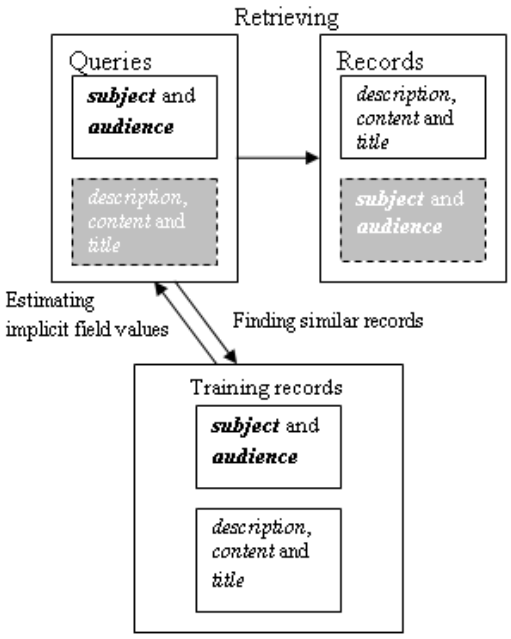


Figure 2.3. Discovering implicit field values for an NSDL search task. Recall that shaded boxes represent *implicit* information.

forward approach – matching query fields against record fields – will yield no relevant results. Our approach will rank testing records by comparing their *title, description* and *content* fields against the query-based SRM. Figure 2.3 depicts our approach for this retrieval task, following the representation of discovering implicit field information for search in Figure 2.1. We point out that in this experiment, we *only* hide the *subject* and *audience* fields in the testing set and the held-out set, and do not hide the other three fields. Furthermore, we do not change any of the five fields in the training set and show that our approach can still build SRM for effective search from training data that contains many empty fields.

We use the standard rank-based evaluation metrics: *precision* and *recall*. Let N_R be the total number of records relevant to a given query, suppose that the first K records in our ranking contain N_K relevant ones. Precision at rank K is defined as $\frac{N_K}{K}$ and recall is defined as $\frac{N_K}{N_R}$. Average precision is defined as the mean precision

over all ranks where relevant items occur. R-precision is defined as precision at rank $K{=}N_R$.

2.5.2.2 Baseline Systems

To demonstrate the advantages of our SRM based approach of discovering implicit field information for retrieval, in experiments we compare the ranking performance of the following retrieval approaches:

cLM is a *cheating* version of un-structured text search using a state-of-the-art language-modeling approach (Ponte and Croft 1998). We disregard the structure, take all query keywords and run them against a *concatenation* of all fields in the testing records. This is a "cheating" baseline, since the concatenation includes the *audience* and *subject* fields, which by our construction are normally missing from the testing records. We use Dirichlet smoothing with parameters optimized on the training data. This baseline mimics the core search capability available on the NSDL website[9].

bLM is a combination of SQL-like structured matching and unstructured search with query expansion. We take all training records that contain an *exact* match to our query and select 10 highly-weighted words from the *title*, *description*, and *content* fields of these records. We run the resulting 30 words as a language modeling query against the concatenation of *title*, *description*, and *content* fields in the testing records. We use this baseline to investigate the performance of using SQL-style exact match and a state-of-the-art query expansion technique – true Relevance Model (Lavrenko and Croft 2001)) – together to address the empty field problem here. This is a non-cheating baseline.

SRM is the Structured Relevance Model. For reasons of both effectiveness and efficiency, we first run the original query to retrieve top-500 records, then use these

[9]http://www.nsdl.org

	cLM	bLM	SRM	%change	improved
Retrieved Relevant:	949	914	861	-5.8	26/50
Average Precision:	0.1790	0.1668	0.2156	29.3	**43/63**
Precision at:					
5 records	0.1651	0.2413	0.3556	47.4	**32/43**
10 records	0.1571	0.2063	0.2889	40.0	**34/48**
20 records	0.1540	0.1722	0.2024	17.5	28/47
R-Precision	0.1587	0.1681	0.2344	39.4	**31/49**

Table 2.6. Performance of the 63 test queries retrieving 1000 records on the testing data. Bold figures show statistically significant differences. Across all 63 queries, there are 1253 relevant records. The *%change* column shows relative difference between SRM and bLM. The Bold figures indicate SRM statistically significantly improves bLM (according to the sign test with $p < 0.05$) in terms of the IR metric in that row.

records to build SRMs. When calculating the cross entropy (Equation 2.9), for each field we only include the top-100 words which will appear in that field with the largest probabilities.

Note that our baselines do not include a standard SQL approach directly on testing records. Such an approach would have perfect performance in a "cheating" scenario with observable *subject* and *audience* fields, but would not match any records when the fields are removed.

2.5.2.3 Results

Table 2.6 shows the performance of the SRM based approach against the two baselines. The model parameters were tuned using the 64 training queries on the *training* and *held-out* sets. The results are for the 63 test queries run against the *testing* set. (Similar results occur if the 64 training queries are run against the *testing* set.) The *%change* column shows relative difference between our model and the baseline bLM. The *improved* column shows the number of queries where SRM exceeded bLM vs. the number of queries where performance was different. For example, 33/49 means that SRM out-performed bLM on 33 queries out of 63, underperformed on 49−33=16 queries, and had exactly the same performance on 63−49=14 queries.

The results show that SRM outperforms the baselines in the high-precision region, beating bLM's mean average precision by 29%. User-oriented metrics, such as R-precision and precision at 10 documents, are improved by 39.4% and 44.3% respectively. These observations indicate: although standard information retrieval techniques and structured field matching could be combined to address the empty field problem, the SRM based approach outperforms them. The absolute performance figures of SRM are also very encouraging. Precision of 28% at rank 10 means that on average almost 3 out of the top 10 records in the ranked list are relevant, despite the requested fields not being available to the model. It is encouraging to see that SRM outperforms cLM, the cheating baseline that takes advantage of the field values that are supposed to be "missing". These results show that our approach can effectively find relevant semi-structured records even when the query-specified fields in those records are empty.

2.5.3 Retrieval Performance of the SRM-based Approach on Data with Different Amount of Missing Information

Now we investigate how our SRM-based retrieval approach performs when encountering different amount of missing information. We design a small-scale *synthetic* experiment where we can control the amount of missing field values and explore the corresponding impact on the retrieval effectiveness of our approach.

We use the NSDL subset described in §2.4 to produce different searched target collections for this experiment. As a reminder, this subset includes 11,596 records that have all five of the *title*, *content*, *description*, *subject* and *audience* fields. The 127 semi-structured queries used in the previous section (§2.5.2) and their corresponding relevance judgments are used again in this experiment. We first divide this NSDL subset into two halves, one for train and one for test. Then we delete the queries

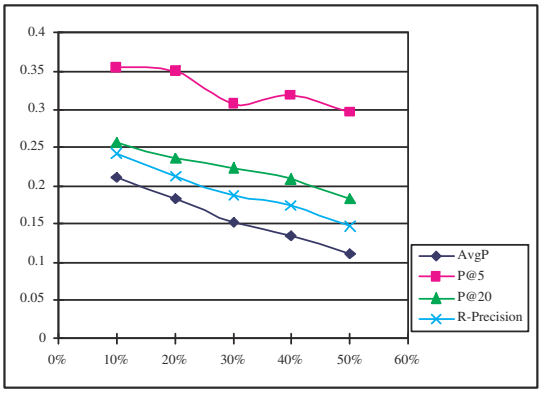
	Most frequent			Most infrequent	
Values dropped	2%	5%	10%	10%	20%
AvgP	0.2497	0.2151	0.1682	0.2822	0.2902
P@5	0.3627	0.3392	0.2843	0.4000	0.4020
P@20	0.2593	0.2309	0.1877	0.2907	0.2917
R-Precision	0.2764	0.2324	0.1786	0.3147	0.3234

Table 2.7. SRM's IR Performance when most frequent or infrequent *description* words are missing.

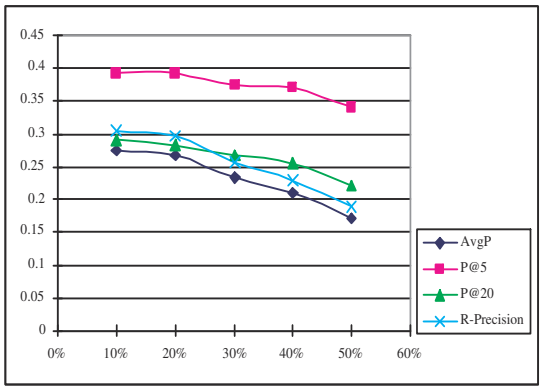
which do not have any relevant records in this NSDL subset – overall we obtain 102 remained queries for this study.

Some experimental setting of the search task here is similar to what we did in §2.5.2. The search goal is to retrieve relevant records from the testing set given a query that contains only *subject* and *audience* fields. All five fields are observable in the training set, while only *title*, *content* and *description* fields are observable (*subject* and *audience* are removed completely) in the testing set. To search relevant records in the testing set, we first use records similar to the given query in the training set to build SRM on the *title*, *content* and *description* fields, then use the built SRM for search. In this experiment, there are overall 3860 relevant records in the testing set for the 102 queries.

We fix all the tuned model parameters in our previous IR experiment in §2.5.2 to investigate the impact of different amount of implicit information on the retrieval performance of our approach. In this study, we simulate the following different missing information situations using the *description* field in both the training and testing sets: (1) some records have empty *description* fields, generated by removing *description* fields from randomly picked records; (2) some records have incomplete *description* fields, generated by removing randomly picked *description* words in each record; (3) all records lose some *description* words that occur most frequently or infrequently in this NSDL subset.



(a) The change of the SRM's IR Performance vs. the percentage of records completely missing *description* fields.



(b) The change of the SRM's IR Performance vs. the percentage of words missing in the *description* field of each record.

Figure 2.4. The impact of the amount of implicit information on the retrieval performance of SRM. Different lines depict different evaluation metrics.

Figure 2.4 (a) and (b) show the change of the retrieval performance with different portion of randomly picked records missing *description* fields and different portion of randomly picked *description* words missing in each record, respectively. Table 2.7 presents the results when the most frequent (top 2%, top 5% or top 10% frequent) or most infrequent (top 10% or top 20% infrequent) *description* words are missing. We use the standard IR evaluation metrics: P@k and R-Precision, used in §2.5.2, for measuring the performance. Figure 2.4 shows that SRM's IR performance degraded little by little when more and more information is missing, i.e. missing part of information will not change SRM's IR performance drastically. This property is very attractive because it enables SRM for more real-world search tasks on semi-structured databases that have different situations of implicit information.

Here we observe that completely deleting *description* fields has a greater impact on the SRM's performance than dropping only part of the *description* fields: when 10% of the information is missing, the average precisions are about 0.21 and 0.27 respectively; when missing 50%, the average precisions are about 0.10 and 0.17 respectively. Furthermore, missing frequent *description* words has a great impact on

SRM's IR performance: when only missing the 10% most frequent values, the average precision degrades to 0.1682. Another interesting observation in this experiment is that when dropping the most infrequent *description* words, the average precision does not degrade at all, e.g. 0.2822 when missing 10% and 0.2902 when missing 20%. However this phenomenon does not mean that we should delete infrequent field values before applying SRM for IR because that this query set is small and biased, which favors the topics appear frequently in the original NSDL snapshot – each query generated is required to have more than 30 relevant records. Therefore, losing more frequent field values has a greater impact on the IR performance on these queries.

2.5.4 Retrieval Experiment on the Monster Dataset

We further employ the Structured Relevance Models to handle implicit field values in another real-world semi-structured record retrieval challenge – matching semi-structured job records and resume records in a large scale job/resume collection provided by the Monster Worldwide company.

2.5.4.1 Overview of the Search Task

We are interested in finding resumes that are appropriate matches to a job description, where *appropriate* means that a prospective employer would be interested in reading the retrieved resumes. Prospective employees or employers usually submit their resume or job information through online forms that contain many free text fields such as *job title*, *biography*, etc. This information is typically maintained by a relational database engine. An ideal system would retrieve candidate resumes for a job or a list of jobs potentially suitable for a candidate. However using a relational engine for this matching task will run into two major obstacles similar to the ones we met when handling the NSDL collection. First, many fields are input as free form text by users rather than a set of agreed upon keywords from a closed vocabulary. That means that the contents cannot be reliably predicted; thus, the problem is more

	records covered	average length	unique terms
ResumeTitle	1,276,566	3	92,403
ResumeBody	988,107	477	1,636,980

Table 2.8. Statistics for some textual fields.

of a classic information retrieval one. Second, many fields are empty: users often do not input all the fields in an online form. For example, in our collection, 23% of the resumes do not have a *ResumeBody* field and 90% of the *Summary* field are empty.

We are given a collection of semi-structured resumes R, a collection of semi-structured jobs J, and some known matched resume/job pairs $< \mathbf{r}, \mathbf{j} >$. The search task is to retrieve a list of related resumes for any existing or new job $\mathbf{j}$, or retrieve a list of related jobs for any existing or new resume $\mathbf{r}$. We focus on the former task here; the latter one can be done in similar way and has similar performance.

2.5.4.2 Data and Methodology

This experiment is performed on a challenging large scale real-world semi-structured collection. Each resume or job is represented as a record that may have some empty fields – i.e. some fields are NULL. Fields can be numeric or textual. In total, the collection contains 1,276,573 resume records (spanning 90 fields, 12 of them textual), 206,393 job records (spanning 20 fields, 9 of them textual) and 1,820,420 resume/job pairs, which are *implicitly* annotated by recording job agents' clicks when the agents were browsing the resumes in the collection and may indicate potentially interesting resumes for a particular job. Table 2.8 shows some statistics for resume fields.

We consider two baselines to demonstrate the advantages of the SRM based approach for this task. In the first baseline, we strip the structure from the resume and job records by concatenating the free form text in all the fields, then run each flattened job record as a query against the flattened resumes using a query likelihood approach. We call this simple language modeling approach "**sLM**". We expect its performance to be weak because it does not have any way to bridge the vocabulary

divide between job descriptions and resumes. For example, if a job has "DB Administrator" in the *JobTitle* field, it is likely that this phrase does not appear in an appropriate candidate's resume's *ResumeBody* field, which may contain "SQL server" or "MySQL" instead that indicate specific computer skills of the candidate. In the second baseline, we also strip the structure of records but leverage past browsed resume'/job pairs in a type of supervised query expansion. This approach is a variation of Relevance Models (Lavrenko and Croft 2001) where the relevance model is built from known relevant documents (resumes) rather than from highly ranked ones. We call this approach **tRM** for "true relevance model." It runs in three steps: (1) we run the flattened job record as a query against the flattened job collection, and retrieve a list of similar jobs; (2) we utilize the resumes that are known (by our relevance judgments) to be related to those retrieved jobs, and build a relevance language model from them; and (3) we run the relevance model against the flattened resume collection and retrieve a list of similar resumes. Note that this approach has the opportunity to bring resume-specific language that is related to the job into the query.

We compare the two baselines above with our SRM based approach. In this approach, we first run *each* field of a given job as a query against the corresponding field of the semi-structured job collection, and merge the field-specific retrieved jobs using weighted cross-entropy (Lafferty and Zhai 2001). We retain only the top k most highly ranked jobs and then use the resumes known to be related to these jobs to build the SRM. Then, we run the built SRM as a query and then rank all resumes according to their similarity, again weighted cross-entropy. For SRM, we use the title and body fields from both resumes and jobs (even though they have the same name, the content is rarely similar). Figure 2.5 depicts our SRM based approach for this retrieval task, following the representation of discovering implicit field values for search in Figure 2.1.

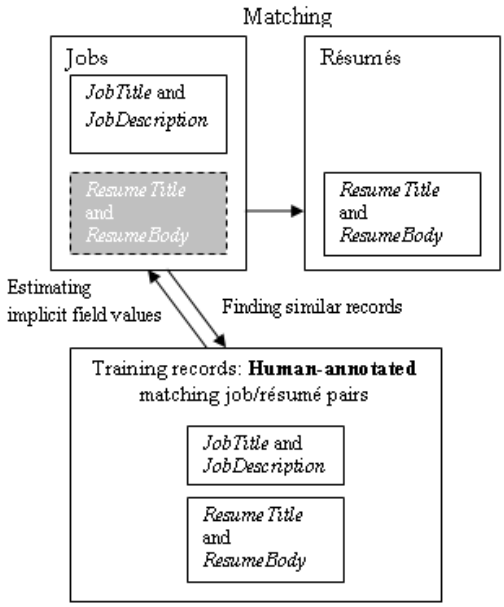


Figure 2.5. Discovering implicit field values for the Monster job/resume matching task. Recall that the shaded box represents implicit information.

In the experiment we first randomly select a set of 300 jobs that had 60-80 annotated matching resumes. We split that set into two halves, one for train and one for test. In addition, we split the set of resumes equally into training and test sub-collections. We use the training resumes to build relevance model or SRM for searching matching resumes in the test collection.

2.5.4.3 Results

Table 2.9 shows the performance of SRM against the two other approaches. We are matching 150 *test* jobs against the *test* resume collection. We use the rank based evaluation metrics (e.g. $P@k$ and R-Precision) here, which have been described in §2.5.2.1 and used in the NSDL retrieval experiments. The *%change* column shows relative difference between SRM and tRM. The *improved* column shows the number of matches where SRM exceeded tRM vs. the number of matches where performance was different.

	sLM	tRM	SRM	%change	improved
Retrieved Matching records:	242	1134	1255	**10.7**	74/116
Average Precision	0.0018	0.0638	0.0726	**13.9**	**101/147**
Precision at:					
5 docs	0.0093	0.1627	0.1947	**19.7**	**23/33**
10 docs	0.0073	0.1460	0.1740	**19.2**	**31/41**
20 docs	0.0070	0.1113	0.1280	**15.0**	**40/58**
R-Precision	0.0055	0.0824	0.0963	**17.0**	**52/68**

Table 2.9. Performance of matching 150 test jobs to the test resume collection. Evaluation is based on retrieving 1000 resumes. Bold figures indicate SRM statistically significantly improves tRM (according to the sign test with $p < 0.05$) in terms of the IR metric in that row. Across all 150 test jobs, there a total of 5173 matched resumes.

The results show that without doing cross-field term inference, a classic retrieval approach such as sLM performs very poorly for this task, i.e. we cannot directly use text from job fields to find matching resumes due to the different languages used in them. By using information from resumes related to a job query, both tRM and SRM achieved much better performance. The tRM approach achieves promising performance by incorporating a form of true relevance feedback to use related resume information in the annotated job/resume pairs. However SRM outperforms tRM by discovering implicit information in each related resume field for retrieval, beating tRM's mean average precision by almost 14%. R-precision and precision at 10 are improved by 17% and 19% respectively.

We note that performing this resume/job matching task on a large-scale real-world semi-structured database is very difficult. At 5 resumes retrieved, the precision of SRM is less than 20% while on average there are 35 annotated training resumes per job (half of the 60-80): that means that on average only 1 of the 35 relevant resumes is found in the top five. To explore each *test* job's matching result further, we categorized the 150 jobs into 3 groups according to precision at 10; the size of each group is shown in Table 2.10. For some jobs, both SRM and tRM find more than 5 matched resumes in the top 10 listed (i.e., P@10 is more than a half). By

P@10	< 0.1	0.1-0.5	> 0.5
SRM	77	49	24
tRM	87	45	18

Table 2.10. Counts of matching resumes' results broken down by P@10 ranges.

looking into the text of some failed matching cases directly we observe that judgments based on click-based implicit annotations are still not good enough. Although still more analysis is needed, these preliminary results demonstrate that SRM is a very promising technique for this challenging task.

2.6 Conclusions

In this chapter, we employed our general perspective of discovering implicit information for search in Figure 2.1(a) to handle implicit field values in searching semi-structured records. We developed a language modeling based technique (called Structured Relevance Models or SRM) which discovers plausible implicit field values in large-scale semi-structured data for retrieval purpose. This technique is based on the idea that plausible values for a given field could be inferred from the context provided by the other fields in the record. To do the inference, SRM follows a generative paradigm and leverages a *hypothetical* probabilistic procedure of generating semi-structured records. At the intuitive level SRM discovers plausible field values for a record using its similar records' corresponding information.

We validated the inference capability of SRM by examining the quality of its discovered field values for *simulated* incomplete semi-structured records in two synthetic experiments. In the first experiment where the inference task is to predict multiple missing *subject* words for records in a small-scale subset of the NSDL collection, the SRM approach performed effectively and achieved lower prediction error rates than a state-of-the-art discriminative learning technique – SVM on the major portion of unique *subject* words that appear less frequently in the collection. In the second large-

scale experiment where the inference task is to select a set of appropriate plausible *subject* words from the whole NSDL collection for simulated incomplete records, SRM also achieved very promising results: it brought 5-6 correct subject words into the top 8 and achieved an average precision of 74.5% for suggesting the subject words. These results demonstrated the effectiveness of using SRM to do cross-field inference in semi-structured data and showed the promising potential of employing SRM for retrieval tasks on large-scale incomplete semi-structured data.

Then we presented how to use SRM to search semi-structured databases that contain incomplete or empty field records. We validated the SRM based retrieval approach with two different large-scale real-world semi-structured data search tasks that involve the empty field problem.

The first search task was performed on a large archive of the NSDL repository. We developed a set of semi-structured queries that had relevant documents in the test portion of collection. We then indexed the test records *without* the fields used in the queries. As a result, using standard field matching approaches, not a single record would be returned in response to the queries—in particular, no relevant records would be found. We showed that standard information retrieval technique and structured field matching could be combined to address the empty field problem, but that the SRM based approach outperforms such an approach. SRM brought two relevant records into the top five—again, querying on empty fields—and achieved an average precision of 22%, a more than 30% improvement over a state-of-the-art relevance model approach combining the structured field matching.

The second search task was performed on a large online job/resume collection. The search goal is to find appropriate resumes matching to a job description. A large set of human-annotated job/resume pairs were used for evaluation. We showed that directly using text from job fields to search matching resumes led to poor results due to the vocabulary mismatch between job fields and resume fields. The retrieval ap-

proaches including SRM and a state-of-the-art relevance model approach performed much better by inferring likely related resume field values for a job query. SRM brought about one matching resume into the top five even the click-based matching judgments are very incomplete. Moreover, by discovering implicit resume field information, SRM outperformed the relevance model approach that does not use the field structure information, achieving 17% and 19% improvement over R-precision and P@10, respectively. We point out that in this search task, we used the real-world data without changing any implicit information in them – this is different from that in the first NSDL search task we *artificially* generated a searched collection where the query-specified fields are forced to be hidden in all searched records.

We further investigated how SRM's IR performance changes when encountering different amounts of missing field values, using a controlled small-scale synthetic retrieval experiment on an NSDL subset collection. Experimental results showed that although SRM's IR performance degraded with more information missing, the performance degraded gradually, i.e., partially losing some field information did not change SRM's IR performance drastically. This property makes the SRM based retrieval approach very attractive for many other real-world search tasks on semi-structured databases where the situation of missing field values may be severe in different degrees.

CHAPTER 3

DISCOVERING IMPLICIT ANCHOR TEXT INFORMATION FOR WEB SEARCH

3.1 Introduction

In this chapter we leverage web anchor text to improve web search. We start with a detailed description of the background of this research issue.

There exist rich dynamic human-generated hyperlink structures on the web. Most web pages contain some hyperlinks, referred to as *anchors*, that point to other pages. Each anchor consists of a destination URL and a short piece of text, called *anchor text*. Anchors play an important role in helping web users quickly and conveniently navigate the web for information they are interested in. Although some anchor text only functions as a navigational shortcut which does not have direct semantic relation to the destination URL (e.g.,"click here" and "next"), it is common that anchor text provides some succinct description of the destination URL's content, e.g. "WWW2010" and "The Nineteenth International WWW Conference" are from some anchors linked to `http://wwwconference.org/www2010/`. Anchor texts are usually reasonable queries that web users may issue to search for the associated URL and have been used to simulate plausible web queries relevant to the associated web pages (Nallapati *et al.* 2003). Dang and Croft (2009) demonstrated that using anchor text to help reformulate user-generated queries can achieve very similar retrieval effectiveness, compared with using a real query log. Therefore, anchor text is highly useful for bridging the lexical gap between user-issued web queries and the relevant web pages. It is arguably the most important piece of evidence used in web ranking functions (Metzler *et al.*

	GOV2	ClueWeb09-T09B
# of web pages	25,205,179	50,220,423
# of inlinks	37,185,508	209,219,465
# of pages having inlinks	376,121 (1.5%)	7,640,585 (15.2%)
# of pages having original or enriched in-links (METZLER *et al.* 2009)	977,538 (3.9%)	19,096,359 (38.0%)

Table 3.1. Summary of in-link statistics on two TREC web corpora used in our study.

2009) and has been widely used in hypertextual domain search tasks like wiki search (DOPICHAJ *et al.* 2009; GEVA 2008) and web search (CRASWELL *et al.* 2001; DOU *et al.* 2009; EIRON and MCCURLEY 2003).

However, previous research has shown that the distribution of the number of in-links on the web follows a power law (BRODER *et al.* 2000), where a small portion of web pages have a large number of in-links while most have few or no in-links. Thus, most web pages do not have in-link associated anchor text, a situation originally referred to as the *anchor text sparsity problem* by Metzler et al. (2009). This problem presents a major obstacle for any web search algorithms that want to use anchor text to improve retrieval effectiveness. Table 3.1 shows the anchor text sparsity problem in two large TREC[1] web corpora (GOV2[2] and ClueWeb09-T09B[3]).

To address this problem, Metzler et al. (2009) proposed *aggregating*, or *propagating*, anchor text across the web hyperlink graph so that web pages without anchor text can be *enriched* with their linked web pages' associated anchor text. Figure 3.1 illustrates the procedure they used to enrich a given web page P_0's anchor text representation in the TREC GOV2 collection. P_0's *original anchor text* $A_{orig}(P_0)$ comes from all anchors that are *directly* linked to P_0 in the web pages external to P_0's site (denoted as $Linked_{ext}(P_0)$). For example, $A_{orig}(P_0)$ consists of anchor text

[1] `http://trec.nist.gov/`

[2] `http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm`

[3] `http://boston.lti.cs.cmu.edu/Data/clueweb09/`

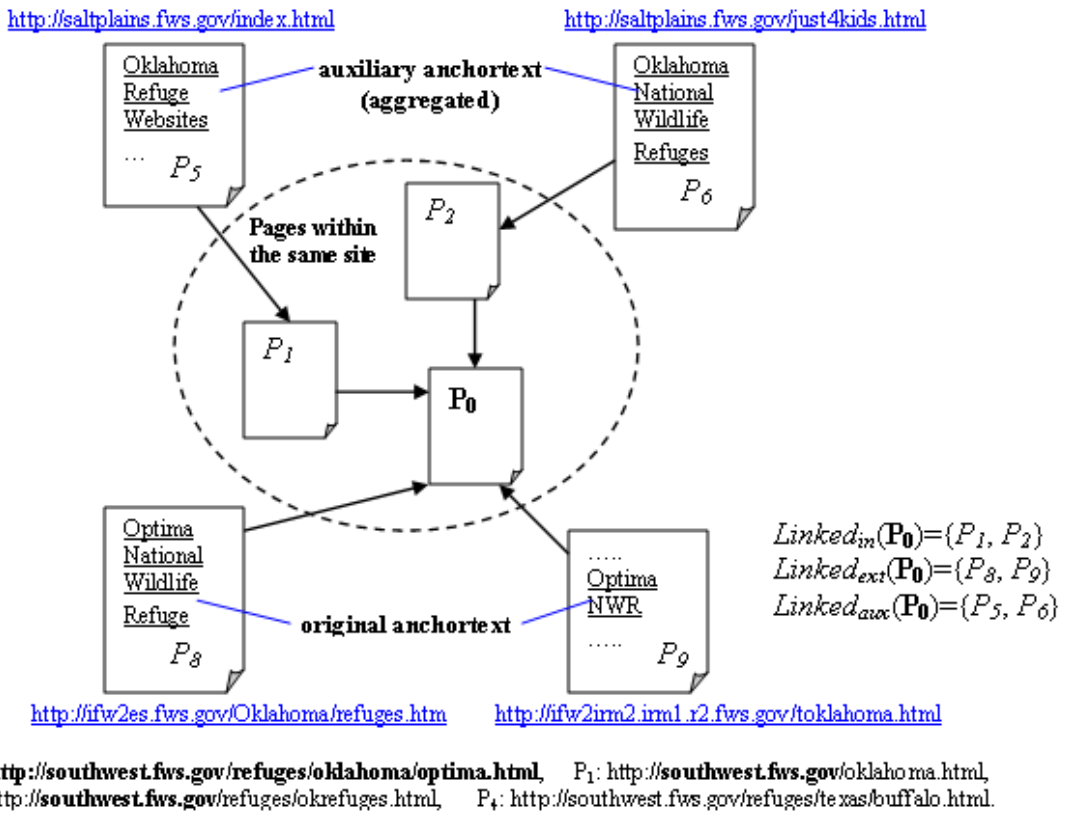


Figure 3.1. Illustration of how to aggregate anchor text over the web graph for discovering plausible additional anchor text for a web page (P_0 in this example). The page P_0 is a GOV2 web page, whose DocID is GX010-01-9459902 and URL is http://southwest.fws.gov/refuges/oklahoma/optima.html.

"Optima National Wildlife Refuge" (in P_8) and "Optima NWR" (in P_9) in Figure 3.1. To enrich P_0's anchor text representation, their procedure first collects all pages $Linked_{in}(P_0) = \{P_1, P_2\}$, within the same site (domain), that link to P_0. Then the procedure collects all anchor text from pages $Linked_{aux}(P_0) = \{P_5, P_6\}$ that are linked to any page in $Linked_{in}(P_0)$ from outside the site. The collected anchor text set $A_{aux}(P_0)$ is used as *auxiliary anchor text*, or *aggregated anchor text*, for enriching P_0's anchor text representation. $A_{aux}(P_0)$ contains anchor text "Oklahoma Refuge Websites" (in P_5) and "Oklahoma National Wildlife Refuges" (in P_6) in this example. The intuition behind Metzler et al.'s approach is that by semantic transition, the original anchor text of the web neighbors may contain good descriptors of the target page.

Metzler et al.'s approach (2009) achieved 38% reduction of URLs with no associated anchor text in a Yahoo! proprietary test web collection. Table 3.1 shows that the number of URLs associated with some anchor text (A_{orig} or A_{aux}) in the two TREC web corpora has also been significantly increased by using their approach. Nevertheless, in Table 3.1 we notice that large portion of web pages still do not have any associated anchor text using their link-based enriching approach. This observation motivated us to consider a content based approach, which does not have specific link structure requirements on the target web page, to further reduce anchor text sparsity and help web search tasks.

Our content based approach to address anchor text sparsity is shown in Figure 3.2(a), following our general perspective of Figure 1.1 (in Chapter 1). We hypothesize that web pages that are similar in content may be pointed to by anchors having similar anchor text due to the semantic relation between anchor text and page content. Under this assumption, we develop a language modeling based technique for discovering a web page's plausible additional anchor text by using anchor text associated with its similar pages. We then use the discovered information for retrieval. Because we

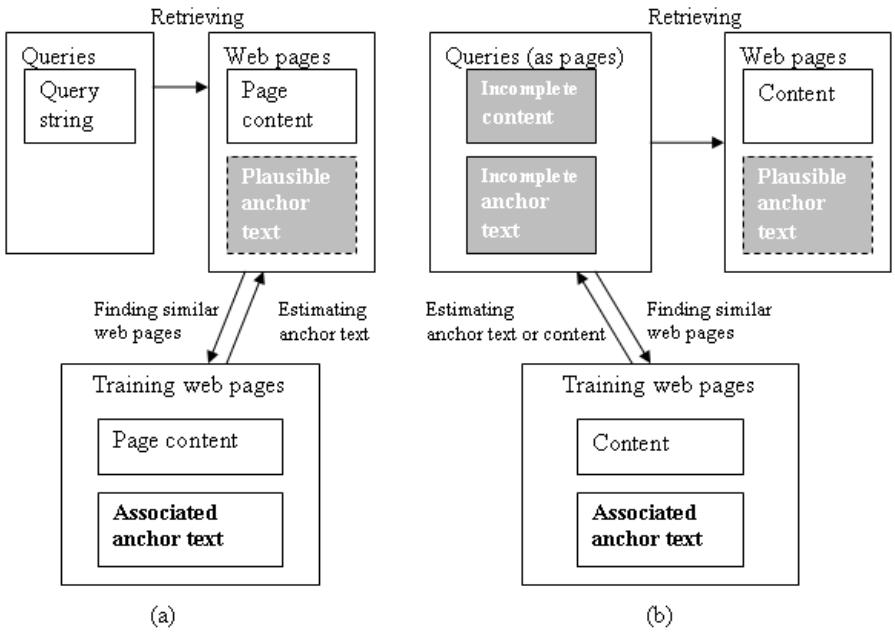


Figure 3.2. The specific perspective of discovering plausible anchor text for web search: (a) using similar web pages for anchor text discovery; (b) viewing queries as web pages and reconstructing better queries for search.

are also interested in using a query-side approach for this task, we examine another retrieval approach depicted in Figure 3.2(b) that naturally emerges from our general perspective of Figure 1.1 (in Chapter 1). Intuitively, this approach adds structure to an unstructured web query and attempts to directly discover the implicit information in the query fields by using the Structured Relevance Model (SRM) approach (described in §2); then the extended semi-structured query is used for retrieval.

We evaluate the performance of the above three different anchor text discovery approaches, including Metzler et al.'s link-based approach, the content approach and the query-side SRM approach, using the named-page finding tasks in the TREC Terabyte tracks (Büttcher *et al.* 2006; Clarke *et al.* 2005).

The remaining parts of this chapter are organized as follows. We begin by reviewing related work in §3.2. Next, in §3.3, we describe webpage-side approaches for discovering anchor text to enrich document representations from §3.3.1 to §3.3.3, and

then directly evaluate the discovered anchor terms by different approaches in §3.3.4. After that, in §3.4 we present how to use different anchor text discovery approaches for web search – we first present language modeling based retrieval models that leverage web pages' discovered anchor text information in §3.4.1; then we formally describe the query-side approach of discovering implicit anchor text information for retrieval in §3.4.2. In §3.4.3 we compare the retrieval performance of different approaches, including both the webpage-side and the query-side, using the named-page finding tasks in the TREC Terabyte tracks. We conclude in §3.5.

3.2 Related Work

Metzler et al. (2009) first directly addressed the anchor text sparsity problem by using the web hyperlink graph and propagating anchor text over the web graph. Our work also addresses the same problem but uses a different approach, which is based on the content similarity between web pages. Our approach is related to other similarity based techniques, such as cluster-based smoothing from the language modeling framework (Kurland and Lee 2004; Kurland and Lee 2006; Liu and Croft 2004; Tao *et al.* 2006), except we focus on enriching web documents' anchor text representation by using their similar documents' associated anchor text.

Anchor text can be modeled in many different ways. Westerveld et al. (2001) and Nallapati et al. (2003) model anchor text in the language modeling approach and calculate an associated anchor text language model to update the original document model for retrieval. Fujii (2008) further considers weighting each piece of anchor text from each anchor pointing to the same page, in order to obtain a more robust anchor text language model. Here, we also adopt the language modeling approach but focus on discovering a plausible associated anchor text language model for web pages with no or few inlinks. Our approach can be easily used together with any

language modeling based retrieval model that takes document structure into account (e.g., Ogilvie and Callan's model (2003)).

Our approach of overcoming anchor text sparsity stems from ideas in the relevance based language models (RMs), proposed by Lavrenko and Croft (2001). Their original work introduces RMs to find plausible useful query expansion terms. Here we adapt the RMs to compute a web content dependent associated anchor language model for positing anchor terms and using anchor text for retrieval. Our approach is also related to an effective contextual translation approach of finding term relations (used for mining related query terms in query logs) in Wang and Zhai's work (2008).

In addition, by viewing anchor text as a special semi-structured textual field of a web page and plugging the same structure into web queries, we can adapt the Structured Relevance Model approach (Lavrenko *et al.* 2007) described in Chapter 2 to do a query-side information discovery for this IR challenge. From a high level point of view, the relation between the web content based approach and the SRM approach is similar in spirit to the relation between document expansion (Liu and Croft 2004; Tao *et al.* 2006) and query expansion (Lavrenko and Croft 2001).

Our content-based approach of discovering anchor terms for web search has also been presented in a published paper (Yi and Allan 2010).

3.3 Discovering Implicit Anchor Text for Web Pages

We now describe three different approaches of discovering plausible anchor text for web pages with few or no inlinks. The goal of each is to produce a ranked list of plausible anchor text terms for a page.

We are interested in the effectiveness of different approaches for discovering plausible description terms for a target page to reduce anchor text sparsity. Thus, we directly evaluate the quality of discovered anchor terms by different approaches in this section. We will evaluate using discovered terms for retrieval later in §3.4.

3.3.1 Aggregating Anchor Text over Web Link Graph

As described in §3.1, Metzler et al.'s approach (2009) collects all the original anchor text pointing to the within-domain web neighbors $Linked_{in}(P_0)$ of a target web page P_0, to discover P_0's plausible additional anchor text. The collected anchor text is called P_0's auxiliary anchor text $A_{aux}(P_0)$. Note that this anchor text aggregation procedure does not use any anchor text associated with internal inlinks (the links between $Linked_{in}(P_0)$ and P_0), because internal inlinks are typically generated by the owner of the site for navigational purposes and their associating anchor text tends to be navigational in nature (e.g., "home","next page", etc.; refer to their paper (Metzler *et al.* 2009) for more discussions on this issue). We emphasize that we follow them here and do not use the anchor text associated with internal inlinks in any way.

We use two typical methods to rank the relative importance of each anchor term w in the A_{aux}. The first method, denoted as **AUX-TF**, is to use each term w's term frequency $tf_{aux}(w)$ in A_{aux}. The second method, denoted as **AUX-TFIDF**, is to use each term w's $tf_{aux} \cdot idf(w)$ score, computed by multiplying $tf_{aux}(w)$ with w's *idf* score in the web collection. The quality of the discovered anchor term rank lists produced from these two link-based methods implies the effectiveness of using A_{aux} for discovering anchor text. We will compare the output term rank lists from these two methods with that from the content based approach in §3.3.4.

3.3.2 Discovering Anchor Text through Finding Similar Web Pages

As discussed in §3.1, many web pages still cannot obtain any auxiliary anchor text by using the link-based approach due to the link structure of them and their web neighbors. Therefore, we propose a different, content based, approach to discover a web page's plausible anchor text. Intuitively, our approach assumes that similar web pages may be described by similar anchor text. For example, in Figure 3.3, the target

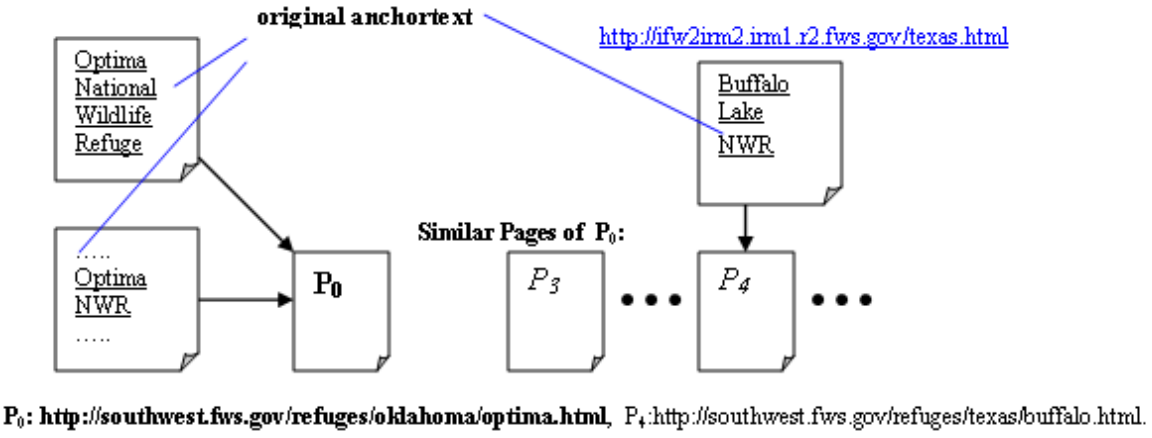


Figure 3.3. Illustration of how to discovering plausible additional anchor text for a web page (P_0 in this example) using its similar pages. The page P_0 is the same GOV2 web page in Figure 3.1.

page P_0 (the same target page in Figure 3.1), which is about Optima national wildlife refuge, is similar in content with the page P_4, which is about Buffalo Lake national wildlife refuge. We observe that the anchor term "NWR", which is the acronym of "national wildlife refuge" and appears in original anchor text $A_{orig}(P_0)$ and $A_{orig}(P_4)$ can be used to partially describe both P_0 and P_4 although two pages are concerned about different places. Note from Figure 3.1 that "NWR" does not appear in auxiliary anchor text $A_{aux}(P_0)$ of the target page P_0.

We consider a language modeling approach to better use document similarity and anchor text information, based on the idea from the relevance-based language model (RM) (Lavrenko and Croft 2001). Given a query q, RM first calculates the posterior $p(D_i|q)$ of each document D_i in the collection $\mathcal{C}$ generating the query q, then calculates a query dependent language model $p(w|q)$:

$$p(w|q) = \sum_{D_i \in \mathcal{C}} p(w|D_i) \times p(D_i|q), \tag{3.1}$$

where w is a word from the vocabulary $\mathcal{V}$ of $\mathcal{C}$. Similarly, given a target page P_0, our approach aims to calculate a relevant anchor text language model (RALM) $p(w|A_0)$

by:

$$p(w|A_0) = \sum_{A_i \in \mathcal{A}} p(w|A_i) \times p(A_i|A_0), \tag{3.2}$$

where A_i denotes the complete original anchor text that should be associated with P_i but which may be *missing*, $\mathcal{A}$ denotes the complete original anchor text space for all pages, and $p(w|A_i)$ is a multinomial distribution over the anchor text vocabulary $\mathcal{V}_\mathcal{A}$.

To compute $p(A_i|A_0)$ in Equation 3.2 where A_0 and A_i information may be incomplete, we view each page P_i's content as its anchor text A_i's context and use P_i's document language model $p_i = \{p(w|P_i)\}$ as A_i's contextual language model (or contextual model). Then we can calculate a translation model $t(A_i|A_0)$ by using A_0 and A_i's contextual models and use $t(A_i|A_0)$ to approximate $p(A_i|A_0)$. This contextual translation approach is also used by Wang and Zhai (2008) for mining related query terms in query logs.

When calculating a page P_i's document language model $p_i = \{p(w|P_i)\}$, we employ Dirichlet smoothing (Lafferty and Zhai 2001) on the maximum likelihood (ML) estimate of observing a word w in the page ($p_{ML}(w|P_i)$), i.e. $p_{ML}(w|P_i)$ is smoothed with the word's collection probability $p(w|\mathcal{C})$ by:

$$p(w|P_i) = \frac{N_{P_i}}{N_{P_i} + \mu} p_{ML}(w|P_i) + \frac{\mu}{N_{P_i} + \mu} p(w|\mathcal{C}), \tag{3.3}$$

where N_{P_i} is the length of P_i's content and μ is the Dirichlet smoothing parameter ($\mu = 2500$ in our experiments). Then given two pages P_0 and P_i, we use the Kullback-Leibler divergence (KL) $Div(\cdot||\cdot)$ between their document models p_0 and p_i to measure their similarity and view it as the contextual similarity between the associated anchor text A_0 and A_i. Then the contextually based translation probability $t(A_i|A_0)$ is calculated by:

$$t(A_i|A_0) = \frac{\exp(-Div(p_0||p_i))}{\sum_i \exp(-Div(p_0||p_i))}. \tag{3.4}$$

This $t(A_i|A_0)$ is then used to approximate $p(A_i|A_0)$ in Equation 3.2 to get:

$$p(w|A_0) \approx \sum_{A_i \in \mathcal{A}} p(w|A_i) \times t(A_i|A_0). \tag{3.5}$$

A few transformations of Equation 3.4 can obtain:

$$t(A_i|A_0) \propto \prod_w p(w|P_i)^{p(w|P_0)}, \tag{3.6}$$

which is the likelihood of generating A_0's context P_0 from A_i's context P_i's smoothed language model and being normalized by A_0's context length. This likelihood can be easily obtained by issuing P_0 as a long query to any language model based search engine. In addition, we use the observed *incomplete* original anchor text language model $p_{obs}(w|A_i)$ associated with P_i to approximate $p(w|A_i)$ in Equation 3.5, and let $p_{obs}(w|A_i) = 0$ if P_i has no $A_{orig}(P_i)$. In this way, the RALM $p(w|A_0)$ can be computed.

In practice, for efficiency the RALM of the target page P_0 is computed from P_0's top-k most similar pages' associated original anchor text because $t(A_i|A_0)$ in Equation 3.4 is very small for the other pages. Due to the anchor text sparsity, we select a surprisingly high $k = 2000$ in our experiments. Because some of these similar pages do not have associated A_{orig}, we use another parameter, m, to denote the number of most similar pages that have original anchor text and so contribute information in the RALM, and we tune m in the experiments. Intuitively, increasing m can increase the number of anchor text samples to better estimate RALM but may also introduce more noise when the sample size is large.

The probability $p(w|A_0)$ of an anchor term w in the RALM directly reflects the goodness of the term w used as original anchor text for the page P_0, thus we use the anchor terms that have the largest probabilities $p(w|A_0)$ in the RALM to evaluate the effectiveness of our content based approach. Theoretically our approach can associate

any web page with some anchor term information if there is some anchor text in the corpus, completely independent of link structure thus further reducing the anchor text sparsity.

3.3.3 Using Keywords as Anchor Text

The keyword based approach comes from the intuition that important keywords in a web page may in and of themselves be good description terms for the page, thus may be arguably used as if they were anchor text. We use two typical term weighting schemes to identify the keywords and rank the words in a web page's content. The first method, denoted as **DOC-TF**, uses each word w's term frequency $tf_{P_0}(w)$ in the page P_0 for term weighting. The second method, denoted as **DOC-TFIDF**, uses each word w's $tf_{P_0} \cdot idf(w)$ score, computed by multiplying $tf_{P_0}(w)$ with w's *idf* score in the web collection. The top ranked terms in a page P_0 by two methods are used as the possible anchor terms for P_0. We will use these two keyword based methods as baselines in the next section.

3.3.4 Evaluating Discovery

We now directly compare the anchor text terms found by different approaches, including two link based methods (AUX-TF and AUX-TFIDF), our content based approach (RALM), and two keyword based methods (DOC-TF, DOC-TFIDF), in order to evaluate the *potential* of using discovered information for retrieval. We will evaluate retrieval directly in §3.4.

3.3.4.1 Data and Methodology

We use two publicly available large TREC web collections (GOV2 and ClueWeb-T09B). GOV2 is a standard TREC web collection (Büttcher *et al.* 2006) crawled from government web sites during early 2004. The ClueWeb09 collection is a much larger and more recent web crawl, which contains over 1 billion pages crawled during

01/06/2009-02/27/2009. ClueWeb09-T09B is a subset of ClueWeb09 and contains about 50 million English web pages. Compared with GOV2 crawled only from the gov domain, ClueWeb09-T09B is crawled from the general web thus is a less biased web sample; in another aspect, GOV2 contains relatively high quality government web pages thus having less noise than ClueWeb09-T09B. We use both GOV2 and ClueWeb09-T09B in our experiments to show how different approaches perform in web collections that have different characteristics.

The Indri Search Engine[4] was used to index both collections by removing a standard list of 418 INQUERY (Broglio *et al.* 1993) stopwords and applying the Krovetz stemmer. In a separate process, we run the Indri Search Engine's harvestlinks utility on the two collections to collect web page inlinks and raw anchor text information where we do not perform stopping or stemming.

In order to evaluate the quality of discovered anchor text for a web page P_0, we need to have the ground-truth anchor text for P_0, which could be subjective and expensive to obtain through human-labeling. Therefore, we consider an alternative approach and utilize web pages that have non-empty original anchor text A_{orig} to generate the evaluation data. Specifically, we first hide a page P_0's existing $A_{orig}(P_0)$, apply different anchor text discovery approaches on P_0, then compare the discovered anchor text with $A_{orig}(P_0)$, the ground-truth anchor text for P_0.

We need to tread carefully because this way of generating evaluation data is *artificial*. The simulated no-anchor-text web pages may have different properties than the web pages that are truly missing anchor text. For example, a web page that is associated with large amount of anchor text could be a high quality home-page of some popular web portal or a very low quality page pointed to by some link-spam farm, thus different from a typical page that has no anchor text. Nevertheless,

[4] `http://www.lemurproject.org/indri/`

using this automatic data generation procedure enables us to leverage large numbers of web pages to compare the relative performance of different approaches with no human labeling effort. Furthermore, the purpose of this set of experiments is solely to evaluate the *potential* of using plausible anchor text discovered by different approaches for retrieval – if the discovered information is of poor quality, we would have no hope for using it for improving search. We will address retrieval itself in §3.4.

For evaluation, we consider each anchor term in the hidden $A_{orig}(P_0)$ of a web page P_0 as a good description term, or a *relevant* term, for P_0 while terms not in $A_{orig}(P_0)$ as *non-relevant* ones; in this way, we generate term relevance judgments for P_0. Then we employ each different approach to discover a ranked list of plausible implicit anchor terms for P_0 and use the relevance judgments to evaluate the ranked anchor term list. Note that for fair comparison, $A_{orig}(P_0)$ is not used in Equation 3.2 for calculating RALM in the content based approach, i.e., we assume that $p_{obs}(w|A_0) = 0$. In the experiments, we perform stopping on the raw anchor text by removing a short list of 39 stopwords, which includes 25 common stopwords (MANNING *et al.* 2008,p.26) and 14 additional anchor terms[5] that are either common navigational purposed words or part of URLs – it is common that anchor text contains a URL.

We calculate typical IR evaluation measurements including Mean Average Precision (MAP), Mean Reciprocal Rank(MRR), Precision at the number of relevant terms(R-Prec), Precision at K (P@k) and also normalized discounted cumulative gain (NDCG) (JÄRVELIN and KEKÄLÄINEN 2002). For all measurements, a higher number indicates better performance. In the experiments, we are specifically interested in the quality of top ranked discovered anchor terms; thus, we only use the top-20 discovered terms to calculate the measurements [6]. In order to compare dif-

[5] *http, https, www, gov, com, org, edu, net, html, htm, click, here, next, home.*

[6] When the number of relevant terms for a page is larger than 20, the R-Prec and MAP may be under-estimated a little while NDCG may be over-estimated a little, depending on the actual positions of the relevant terms in the term rank lists after the top-20 cut.

	MAP	NDCG	MRR	P@5	P@20	R-Prec	Discovered Rel.
DOC-TF	0.3162	0.4585	0.5441	0.2800	0.1333	0.2716	400
DOC-TFIDF	0.2936	0.4348	0.5400	0.2613	0.1240	0.2530	372
AUX-TF	0.1969	0.2598	0.3707	0.1773	0.0643	0.1643	193
AUX-TFIDF	0.1716	0.2423	0.3442	0.1720	0.0647	0.1428	194
RALM	0.3183	0.4275	0.5050	0.2840	0.1140	0.3051	342

Table 3.2. Performance on the GOV2 collection. There are 708 relevant anchor terms overall. The last column shows overall relevant anchor terms discovered by each different approach. RALM performs statistically significantly better than AUX-TF and AUX-TFIDF by each measurement in columns 2–7 according to the one-sided t-test ($p < 0.005$). There exists no statistically significant difference between each pair of RALM, DOC-TF and DOC-TFIDF by each measurement according to the one-sided t-test ($p < 0.05$).

	MAP	NDCG	MRR	P@5	P@20	R-Prec	Discovered Rel.
DOC-TF	0.3517	0.4891	0.5588	0.2373	0.1090	0.2990	327
DOC-TFIDF	0.3107	0.4388	0.5145	0.2213	0.0983	0.2608	295
AUX-TF	0.1840	0.2507	0.3309	0.1463	0.0577	0.1675	172
AUX-TFIDF	0.1634	0.2347	0.3116	0.1383	0.0560	0.1402	167
RALM	0.2612	0.3615	0.4630	0.1733	0.0770	0.2398	231

Table 3.3. Performance on the ClueWeb09-T09B collection. There are 582 relevant anchor terms overall. The last column shows overall relevant anchor terms discovered by each different approach. DOC-TF performs statistically significantly better than both RALM and AUX-TF by each measurement in columns 2–7 according to the one-sided t-test ($p < 0.05$). RALM performs statistically significantly better than AUX-TF and AUX-TFIDF by each measurement in columns 2–7 according to the one-sided t-test ($p < 0.05$).

ferent approaches including the link-based approach, we randomly sample web pages that have both associated A_{orig} and some auxiliary anchor text A_{aux} collected from the web graph for generating evaluation data. For each of two collections, 150 random samples are used for training and another 150 samples for testing. On each training set from two collections, RALM's parameter $m = 15$ described in §3.3.2 achieves the highest MAP.

3.3.4.2 Results and Analysis

The performance of discovering original anchor text by different approaches on the testing set of GOV2 and ClueWeb-09-T09B are shown in Table 3.2 and Table 3.3,

respectively. The results show that the content based approach (RALM) can effectively discover plausible implicit anchor terms in both collections that have different anchor text sparsity, e.g. RALM's $MRR \approx 0.5$, means the first relevant anchor term is discovered at about the $2nd$ rank position of its term rank list on average, in both collections; R-Prec$\approx$ 1/3 in GOV2 and 1/4 in ClueWeb09-T09B, means about 1/3 or 1/4 portion of top R discovered terms in the corresponding collection are relevant where R is the number of relevant anchor terms that would have ideally been found. Furthermore, on both collections RALM performs *statistically* significantly better than two link based approaches (AUX-TF and AUX-TFIDF), which only use the auxiliary anchor text collected over the web graph, with respect to all measurements. This indicates that, for discovering a page's plausible anchor text, the anchor text associated with the similar pages provides more useful information than that associated with the linked web neighbors. The numbers of discovered relevant anchor terms by different approaches, shown in the last column of two tables, also show that only using auxiliary anchor text misses more original anchor text information than our content based approach.

Another observation is that RALM is not *statistically* significantly better on GOV2 and is worse on ClueWeb09-T09B than the keyword based approaches. This indicates that words having high IR utility (high tf or $tf \cdot idf$ scores) are often also good description terms for the page and could be used by human being as the anchor text. Removing a long list of stopwords from web page content has also helped the keyword based approaches to effectively select good description words from the web content. One plausible reason that RALM performs relatively poorly on ClueWeb09-T09B is that, compared with the high quality GOV2 pages, ClueWeb pages are crawled from the general web, where the inlinks and anchor text may be generated in a more noisy way (e.g. spam), degrading RALM's performance. To better understand the performance of different approaches, in Table 3.4 and Table 3.5 we show the top-

10 words of the anchor term rank lists discovered by different approaches for one evaluation web page in GOV2 and ClueWeb09-T09B, respectively.

Although using keyword information can discover some good anchor terms, the content-generated anchor terms found by the keyword based approaches do not help bridge the lexical gap between a web page and varied queries that attempt to search the page, since the content-generated ones *already exist* in the web page content. In contrast, human generated anchor text is highly useful for reducing the word mismatch problem in web search because the lexical gap between anchor text and real queries is relatively small (Metzler *et al.* 2009). Indeed, anchor text has been used as competitive surrogates of real queries for helping search, such as providing effective query reformulations (Dang and Croft 2009). Here, we examine the anchor terms discovered by different approaches to investigate whether our approach can discover anchor text similar in nature to human generated anchor text thus have the potential to also reduce word mismatch for search.

We first use the overlap number of the terms discovered by different approaches for each web page to calculate some lexical gap size measurements. We use the outputs from the keyword based DOC-TF, the link based AUX-TF, and our content based RALM in this analysis. For each web page i in the testing set, we calculate the intersection number $I_i(X, Y)$ of the discovered terms by the X and Y approaches, then compute the total intersection number $I(X, Y)$ by:

$$I(X, Y) = \sum_i I_i(X, Y). \tag{3.7}$$

In addition, for each page i, we calculate the percentage $pct_i(X, Y)$ of the terms discovered by the X approach also appearing in the ones discovered by the Y approach, then compute the average percentage $pct(X, Y)$ with all the pages.

Table 3.6 and Table 3.7 show the term intersection number $I(X, Y)$ between each pair of three approaches on the GOV2 and ClueWeb09-T09B, respectively. Table 3.8 shows three average percentage ratios $pct(X, Y)$ which we have specific interest in.

"Optima National Wildlife Refuge", "Optima NWR", "Washita Optima National Wildlife Refuge near Butler OK"					
DOC-TF	$tf_{P_0}(w)$	**DOC-TFIDF**	$tf_{P_0}idf(w)$	**AUX-TF**	$tf_{aux}(w)$
refuge ⋆	15	refuge ⋆	79.69	oklahoma	6
wildlife ⋆	10	optima⋆	74.30	wildlife ⋆	2
oklahoma	10	hardesty	47.48	refuge⋆	2
optima ⋆	8	hawk	36.20	website	1
species	6	oklahoma	36.03	u	1
hawk	6	wildlife ⋆	31.98	service	1
habitat	6	guymon	29.35	s	1
area	6	habitat	26.42	office	1
prairie	5	species	23.70	national ⋆	1
national	5	quail	21.74	fish	1
AUX-TFIDF	$tf_{aux}idf(w)$	**RALM**	$P(w\|A_0)$	**Rel.**	
oklahoma	21.62	nwr ⋆	0.1164	**butler**	
refuge ⋆	10.62	wildlife⋆	0.0834	**national**	
wildlife ⋆	6.40	refuge⋆	0.0834	**near**	
fish	3.11	national ⋆	0.0834	**nwr**	
u	3.03	general	0.0657	**optima**	
website	2.36	brochure	0.0657	**refuge**	
office	1.54	kansas	0.0601	**washita**	
s	1.29	lake	0.0522	**wildlife**	
national⋆	1.22	tear	0.0308		
service	1.09	sheet	0.0308		

Table 3.4. Discovered plausible anchor terms and their term weights by applying different approaches on one GOV2 web page (TREC DocID in GOV2: GX010-01-9459902) . The first row shows the original three pieces of anchor text associated with the page. The Rel column in bold font shows the term relevance judgments extracted from the first row. ⋆ indicates the relevant terms in the output lists by each approach according to the Rel column. RALM can discover some term like "NWR" (underlined in the table), which does not appear in both the page and the auxiliary anchor text, thus may help to bridge the lexical gap between pages and web queries as using the original anchor text does.

"Weight Loss Resolutions", "Weight Loss New Year's Resolution to Lose Weight" "Resolve to Lose Weight"					
DOC-TF	$tf_{P_0}(w)$	**DOC-TFIDF**	$tf_{P_0}idf(w)$	**AUX-TF**	$tf_{aux}(w)$
weight ⋆	46	weight ⋆	96.38	weight ⋆	709
loss ⋆	26	loss⋆	78.65	loss ⋆	705
lose ⋆	20	lose ⋆	64.47	diet	32
new ⋆	17	resolution⋆	46.57	weightloss	21
year ⋆	15	diet	34.27	guide	20
resolution	13	goal	26.01	scott	8
time	12	eat	25.61	jennifer	8
make	10	year⋆	23.90	contact	8
goal	9	calorie	15.73	site	6
diet	9	pound	15.34	s ⋆	4
AUX-TFIDF	$tf_{aux}idf(w)$	**RALM**	$P(w\|A_0)$	**Rel.**	
loss ⋆	2132.63	weight ⋆	0.2245	**lose**	
weight ⋆	1485.49	loss ⋆	0.1737	**loss**	
weightloss	157.70	diet	0.0550	**new**	
diet	121.86	easy	0.0436	**resolution**	
guide	37.26	lose ⋆	0.0422	**resolve**	
jennifer	33.96	way	0.0412	**s**	
scott	28.52	myth	0.0396	**weight**	
guidesite	22.04	warn	0.0232	**year**	
em	13.15	ppa	0.0232		
mlibrary	11.37	fda	0.0232		

Table 3.5. Discovered plausible anchor terms and their term weights by applying different approaches on one ClueWeb09 web page (ClueWeb09 RecordID: clueweb09-en0004-60-01628). The first row shows the original three pieces of anchor text associated with the page. The Rel column in bold font shows the term relevance judgments extracted from the first row. ⋆ indicates the relevant terms in the output lists by each approach according to the Rel column. The keyword approaches discovered "new year resolution", which may be hard to be discovered by using the page's web-graph neighbor pages' anchor text or using the page's similar pages' anchor text.

I(X,Y)	DOC-TF	AUX-TF	RALM
DOC-TF	2996	281	530
AUX-TF	281	1358	556
RALM	530	556	2879

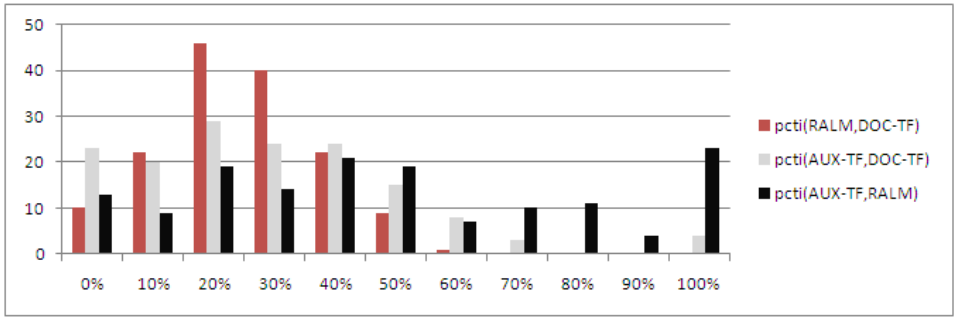
Table 3.6. The intersection number $I(X, Y)$ of the discovered terms between each pair of three approaches on GOV2, where X and Y take each cell value in the first column and row, respectively.

I(X,Y)	DOC-TF	AUX-TF	RALM
DOC-TF	2986	518	667
AUX-TF	518	2028	830
RALM	667	830	2982

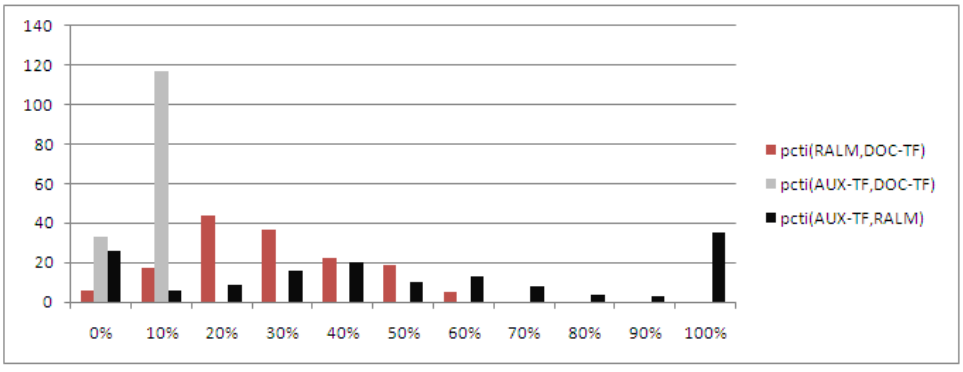
Table 3.7. The intersection number $I(X, Y)$ of the discovered terms between each pair of three approaches on ClueWeb09-T09B, where X and Y take each cell value in the first column and row, respectively.

	GOV2	ClueWeb09-T09B
pct(AUX-TF, DOC-TF)	30.5%	26.0%
pct(AUX-TF, RALM)	47.6%	46.3%
pct(RALM, DOC-TF)	26.0%	22.3%

Table 3.8. The average percentage $pct(X, Y)$ of the terms discovered by the X approach appearing in the ones discovered by the Y approach.



(a)



(b)

Figure 3.4. The number of web pages (Y-axis) with their $pct_i(\cdot,\cdot)$ values falling into the same binned percentage range vs. the binned percentage ranges. (a) results from 150 ClueWeb09-T09B pages; (b) results from 150 GOV2 pages.

Figure 3.4 further shows for each of the three pair in Table 3.8, the count of web pages which have their term overlap per page ratios $pct_i(X, Y)$ between the (X,Y) approaches falling into each binned percentage range in the X axis.

We have these observations: (1) RALM and AUX-TF have the largest intersection numbers (556 and 830 in GOV2 and ClueWeb09-T09B, respectively,) of overlapped terms between different approaches; (2) AUX-TF's discovered terms have much higher average overlap ratio $pct(X, Y)$ with RALM's (47.6% and 46.3% in GOV2 and ClueWeb09-T09B, respectively,) than with DOC-TF's (30.5% and 26.0% in GOV2 and ClueWeb09-T09B, respectively); (3) for many individual web pages, AUX-TF's discovered terms have high overlap with RALM's and that RALM's discovered terms have low overlap with the ones generated by DOC-TF.

These results show that compared with the anchor text generated by using the web content's keywords, the anchor text discovered by our approach is much more similar in nature to the auxiliary anchor text, which is also human generated. Therefore, RALM can also be useful to bridge the lexical gap between web pages and queries and help search. Indeed, in later retrieval experiments, we show that RALM can not only bring additional information not in the original web page content for search, but also discover implicit information indicated by the observed anchor text to further improve retrieval performance.

3.4 Using Discovered Information for Web Search

We now use the discovered anchor text information in different retrieval approaches. We first present language modeling based retrieval models that use discovered anchor text of web pages for *reranking* them. This approach of using webpage-side discovered information for retrieval is shown in Figure 3.2(a) in the introduction of this chapter (also shown in §1.3). Then we formally describe a query-side approach (shown in Figure 3.2(b)) of discovering implicit anchor text information for retrieval

in §3.4.2. After that, we evaluate the retrieval performance of different approaches with the named-page finding tasks in the TREC Terabyte tracks.

3.4.1 Retrieval Models based on Document Smoothing

We follow the typical language modeling based retrieval approach (Ponte and Croft 1998) and score each web page P for a query Q by the likelihood of the page P's document language model $p(w|P)$ generating the query Q:

$$p(Q|P) = \prod_{w \in Q} p(w|P). \tag{3.8}$$

When using Dirichlet smoothing, the document language model $p(w|P)$ can be calculated by Equation 3.3 and then used in Equation 3.8 for retrieval. We call this query likelihood baseline **QL**. We fix $\mu = 2500$ in Equation 3.3 for the document models used to calculate RALM, but tune the μ for QL to achieve the best retrieval performance in the retrieval experiments in §3.4.3.

We follow the mixture model approach (Nallapati *et al.* 2003; Ogilvie and Callan 2003) to use the discovered anchor text information for helping retrieval. In this approach, a web page P's document language model is assumed to be a mixture of multiple component distributions where each component is associated with a prior probability, or a mixture weight. Therefore, we can estimate a language model $p(w|A)$ from anchor text discovered by each different approach for the page P and use $p(w|A)$ as a component of P's document model thus obtaining a better document language model $\tilde{p}(w|P)$:

$$\tilde{p}(w|P) = \alpha p(w|P) + (1 - \alpha) p(w|A), \tag{3.9}$$

where $p(w|P)$ is the original smoothed document model in the QL baseline. Then we can plug $\tilde{p}(w|P)$ into Equation 3.8 for retrieval. We compare the retrieval performance of document language models updated by different discovered anchor text information.

We consider three different anchor text sources to update a web page P's document model: (1) the observed original anchor text $A_{orig}(P)$, (2) the auxiliary anchor text $A_{aux}(P)$, and (3) the RALM computed by our approach for P. We estimate the anchor text language model $p(w|A_{orig})$ and $p(w|A_{aux})$ by using the ML estimate of observing each word w in $A_{orig}(P)$ and $A_{aux}(P)$, respectively. Here, we define the following five retrieval methods that use the above three anchor text sources for document smoothing in order to compare the relative utilities of different anchor text discovery approaches for retrieval:

1. **ORG**, which only uses the observed original anchor text language $p(w|A_{orig})$.

2. **AUX**, which only uses the auxiliary anchor text language $p(w|A_{aux})$.

3. **ORG-AUX**, which uses both $p(w|A_{orig})$ and $p(w|A_{aux})$ to update the document model $p(w|P)$ by:

$$\begin{aligned}\tilde{p}(w|P) = \beta(\alpha p(w|P) + (1-\alpha)p(w|A_{orig})) \\ +(1-\beta)p(w|A_{aux}).\end{aligned} \tag{3.10}$$

4. **RALM**, which only uses the RALM $p(w|A_0)$ in Equation 3.2. The original anchor text of P_0 is not used in Equation 3.2 for calculating RALM.

5. **ORG-RALM**, which uses both $p(w|A_{orig})$ and the RALM $p(w|A_0)$ in Equation 3.2 by:

$$\begin{aligned}\tilde{p}(w|P) = \beta(\alpha p(w|P) + (1-\alpha)p(w|A_{orig})) \\ +(1-\beta)p(w|A_0).\end{aligned} \tag{3.11}$$

   The original anchor text of P_0 is not used in Equation 3.2 for calculating RALM.

Note that different from the experiments in §3.3.4, in the retrieval experiments (§3.4.3) we use the estimated probability of every anchor term instead of the top-20

most important terms discovered by different approaches to update the document language model in each retrieval method. In addition, here we do not consider baselines using anchor text generated by the keyword based approaches described in §3.3.3. The reason is that the content-generated anchor text *already exists* in the web page thus will have almost the same retrieval performance as the QL baseline since no additional information is brought into web pages to match query words.

3.4.2 Query-side Implicit information Discovery for Retrieval

Following the general perspective from Figure 1.1, we further consider using a semi-structured approach in Figure 3.2(b) (in the introduction of this chapter) to handle the implicit anchor text information for this IR challenge. The basic idea of this approach is to add structure to the original unstructured web query and then utilize the semi-structured approach described in Chapter 2 to discover the implicit information in the query fields for retrieval.

Formally, we view queries as well as web pages as semi-structured records containing two fields: *Content* (denoted by $\mathbf{w}_c$) and *Associated Anchor Text* (denoted by $\mathbf{w}_a$). Then given a query q, we first generate a semi-structured query $\mathbf{q} = \{\mathbf{w}_c, \mathbf{w}_a\}$ by duplicating the query string in both fields, i.e. $\mathbf{w}_c = \mathbf{w}_a = q$. We assume that both fields are incomplete and then use the Structured Relevance Models (**SRM**) approach (described in §2.3) to estimate plausible implicit query field values. The whole web collection $\mathcal{W}$ (pages and their associated anchor text) are used as training data. Specifically, we calculate a set of relevance models $\{R_c(\cdot), R_a(\cdot)\}$ for $\mathbf{q}$, where the relevance model $R_i(w)$ specifies how plausible it is that the word w would occur in the field i of $\mathbf{q}$ given the observed $\mathbf{q} = \{\mathbf{w}_c, \mathbf{w}_a\}$, i.e.

$$R_i(w) = P(w \circ \mathbf{w}_i | \mathbf{q}) = P(w \circ \mathbf{w}_i | \mathbf{w}_c, \mathbf{w}_a), i \in \{c, a\}, w \in \mathcal{V}_i, \tag{3.12}$$

where $w \circ \mathbf{w}_i$ denotes appending word w to the string $\mathbf{w}_i$ and $\mathcal{V}_i$ denotes the vocabulary of the field i. Using the training web page records $\mathbf{w}'$ and Equation 3.12, $R_i(w)$ can be further calculated by:

$$R_i(w) = \sum_{\mathbf{w}' \in \mathcal{W}} p(w|\mathbf{w}'_i) \times P(\mathbf{w}'|\mathbf{q}), i \in \{c, a\}, w \in \mathcal{V}_i. \tag{3.13}$$

To calculate the posterior probability $P(\mathbf{w}'|\mathbf{q})$ in Equation 3.13, we use the following equations:

$$\begin{aligned} P(\mathbf{w}'|\mathbf{q}) &\propto P(\mathbf{q}|\mathbf{w}') * P(\mathbf{w}'), \\ P(\mathbf{q}|\mathbf{w}') &= P(\mathbf{w}_c|\mathbf{w}'_c) * P(\mathbf{w}_a|\mathbf{w}'_a) \end{aligned} \tag{3.14}$$

where $P(\mathbf{w}')$ is assumed to be a uniform distribution. In this way, the SRM $\{R_c(\cdot), R_a(\cdot)\}$ for $\mathbf{q}$ can be computed. In practice, for efficiency we do not need to use all records $\mathbf{w}' \in \mathcal{W}$ to calculate $R_i(w)$ in Equation 3.13; instead, we use $\mathbf{q}$'s top-k most similar records (the records that have the top-k largest posteriors $P(\mathbf{w}'|\mathbf{q})$) because $P(\mathbf{w}'|\mathbf{q})$ is small for other records. k is a small number to be tuned by using training queries.

Once we have computed the SRM, we can interpolate it with the original query language model to obtain a better SRM for retrieval:

$$R'_i(w) = \lambda * (p(w|\mathbf{w}_i)) + (1 - \lambda) * R_i(w), i \in \{c, a\}, w \in \mathcal{V}_i, \tag{3.15}$$

which is in spirit the same as the approach called Relevance Model 3 (Diaz and Metzler 2006). The parameter λ allows us to vary the impact of the original query language model on the updated SRM and is tuned by using training queries. Now we can rank web page records $\mathbf{w}' \in \mathcal{W}$ by their similarity to the updated SRM. As described in §2.5.1, we use weighted cross-entropy (Lafferty and Zhai 2001) to measure the similarity between records by:

$$H(R'_{c,a}; \mathbf{w}'_{c,a}) = \sum_{i \in (c,a)} \alpha_i \sum_{w \in \mathcal{V}_i} R'_i(w) \log p(w|\mathbf{w}'_i) \tag{3.16}$$

The outer summation goes over every field of interest, while the inner extends over all the words in the vocabulary of each field i. Meta-parameters α_i allow us to vary the importance of different fields in the final ranking and are tuned by using training queries. This retrieval approach is denoted as **SRM** in the experiments.

In addition, we can also use the query likelihood $P(\mathbf{q}|\mathbf{w}')$ in Equation 3.14 for directly ranking retrieved web pages without discovering implicit anchor text information. This is a structured version of query likelihood retrieval baseline described at the beginning of §3.4.1; thus, we call it **S-QL** in the experiments. In this method, similar to the QL baseline, we employ Dirichlet smoothing for each record field when computing $P(\mathbf{q}|\mathbf{w}')$; different from QL, we tune a different Dirichlet parameter (μ_c and μ_a) for each field. Essentially, S-QL is the first round retrieval in the SRM retrieval method. In practice, in order to adjust the contribution of each field in the final ranking of S-QL, we modify the calculation of the $P(\mathbf{q}|\mathbf{w}')$ in Equation 3.14 by adding some meta-parameters β_c and β_a:

$$P(\mathbf{q}|\mathbf{w}') = P(\mathbf{w}_c|\mathbf{w}'_c)^{\beta_c} * P(\mathbf{w}_a|\mathbf{w}'_a)^{\beta_a}, \tag{3.17}$$

where the roles of β_c and β_a are in spirit similar to that of α_i in Equation 3.16 and tuned by using training queries.

3.4.3 IR Experiments

We use the TREC named-page finding tasks in the Terabyte Tracks (Büttcher *et al.* 2006; Clarke *et al.* 2005) to evaluate the performance of different retrieval methods described in previous sections. The objective of the named page (NP) finding task is to find a particular page in the GOV2 collection, given a topic that describes it. We use the NP topics and their relevance judgments for our experiments. In this experiment, we use Porter stemmer and do not remove stopwords when indexing the GOV2 collection.

For each NP query, we first use the simple query likelihood approach to run the query against the GOV2 collection and obtain the QL baseline. After that we employ the five retrieval methods (described in §3.4.1) that use web pages' discovered implicit anchor text for the task. We use the *reranking* approach, which uses each retrieval method to rerank the top-100 web pages returned by QL. Next, we employ two retrieval methods (S-QL and SRM in §3.4.2) that do query-side implicit field information discovery for the task. After we obtain retrieval results from different methods, the results (web page ranked lists) are evaluated by two TREC measurements previously used for this task (CLARKE *et al.* 2005): **MRR** which is the mean reciprocal rank of the first correct answer and the **%Top10** which is the proportion of queries for which a correct answer was found in the first 10 search results.

We use the TREC 2005 NP topics (NP601-872) for training and the TREC 2006 NP topics (NP901-1081) for testing. For the QL baseline, we tune the Dirichlet parameter $\mu = 500$ to achieve the highest MRR on the training set and obtain QL's top-100 web pages for reranking. Then we fix $\mu = 500$ to calculate the smoothed document model component $p(w|P)$ in the five retrieval methods in §3.4.1, but tune the mixture parameters α and β for them to achieve the highest MRRs with the training queries. For the RALM and ORG-RALM method in these five methods, the parameter m of RALM is also tuned. For the S-QL baseline, we tune the Dirichlet parameters $\mu_c = 500, \mu_a = 50$ for the *Content* field and *Associated Anchor Text* field respectively, and also meta-parameters β_c, β_a to achieve the highest MRR on the training set. Then these Dirichlet parameters are fixed in the S-QL to compute the posterior probabilities used for building relevance models in the SRM method. For SRM, we also tune the interpolation parameter λ in Equation 3.15 and the field weights α_i in Equation 3.16 to achieve the highest MRR on the training set. Finally, we fix the tuned parameters of different methods and test them on the test queries.

	MRR	%Top10	Opt. Param.
QL	0.3132	49.7	
Query-side anchor text discovery:			
S-QL	0.3588‡	58.0	$\beta_c = 0.95, \beta_a = 0.05$
SRM	0.3592‡	58.0	$\alpha_c = 0.95, \alpha_a = 0.05, \lambda = 0.99,$ top-50 docs and top-50 terms in each field
Webpage-side anchor text discovery:			
ORG	0.3696‡	57.5	$\alpha = 0.95$
	Link-based approaches:		
AUX	0.3187	50.8	$\alpha = 0.99$
ORG-AUX	0.3711‡	57.5	$\alpha = 0.95, \beta = 0.99$
	Content-based approaches:		
RALM	0.3388‡	53.6	$m = 20, \alpha = 0.95$
ORG-RALM	0.3975$^{\triangle\ddagger}$	59.7	$\alpha, \beta = 0.95, m = 20$

Table 3.9. Retrieval performance of different approaches with TREC 2006 NP queries. The $^{\triangle}$ indicates statistically significant improvement over MRRs of ORG and ORG-AUX and SRM. The ‡ indicates statistically significant improvement over MRRs of QL and AUX. All the statistical tests are based on one-sided t-test ($p < 0.05$).

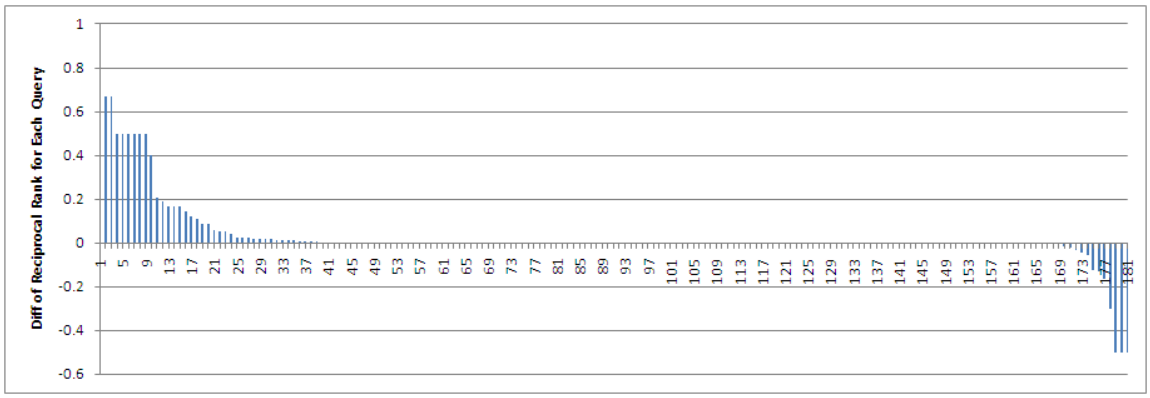


Figure 3.5. The difference of the reciprocal ranks (RR) between ORG-RALM and ORG on each individual NP topic. Above the x-axis reflect queries where ORG-RALM out-performs ORG. Y-axis denotes the actual difference, computed using (ORG-RALM's RR minus ORG's RR) of each NP finding query. All the differences are sorted then depicted to show the IR performance difference of two approaches. Among 181 queries, ORG-RALM outperforms ORG on 39 queries, performs the same as ORG on 126 queries and worse than ORG on 16 queries.

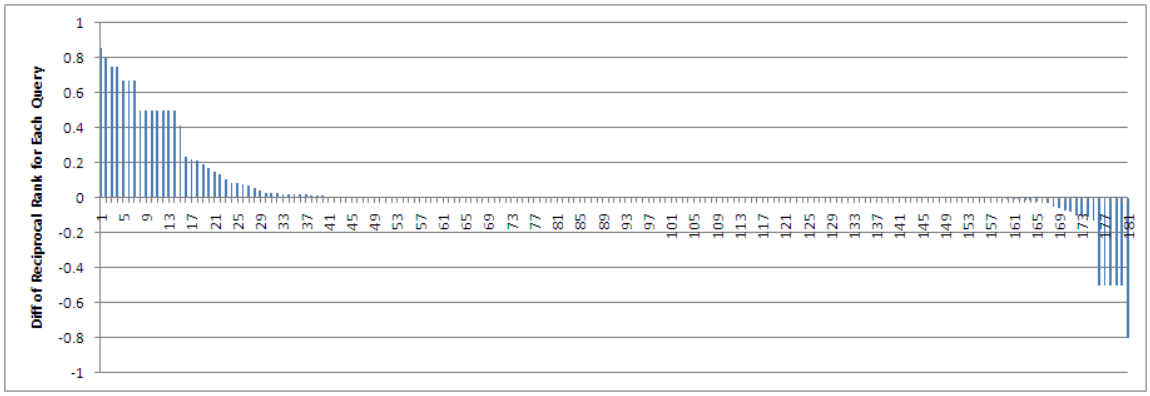


Figure 3.6. The difference of the reciprocal ranks (RR) between ORG-RALM and SRM on each individual NP topic. Above the x-axis reflect queries where ORG-RALM out-performs SRM. Y-axis denotes the difference, computed using (ORG-RALM's RR minus SRM's RR) of each NP finding query. All the differences are sorted then depicted to show the IR performance difference of two approaches. Among 181 queries, ORG-RALM outperforms SRM on 43 queries, performs the same as SRM on 115 queries and worse than SRM on 23 queries.

Table 3.9 shows the retrieval performance of different methods on the test queries and the tuned parameters in each method. Figure 3.5 and 3.6 further show the difference of the reciprocal ranks (RR) between ORG-RALM and ORG, as well as between ORG-RALM and SRM, on each individual NP finding query, respectively. We have the following main observations:

1. S-QL performs similarly to ORG, which uses the original anchor text of web pages and the document smoothing approach, but statistically significantly worse than ORG-RALM, which uses both the original anchor text and the discovered anchor text information for document smoothing. It is not a surprise S-QL performs similarly to ORG because they both *only* use the observed anchor text information for search.

2. SRM performs similarly to ORG but statistically significantly worse than ORG-RALM. ORG-RALM performs differently from SRM on 66 queries (36.5% of all 181 test queries), where ORG-RALM outperforms SRM on 43 of them. One

plausible reason of the relative inferior performance of SRM (compared with ORG-RALM) is that most queries in this search task are navigational queries and it is known that query expansion using pseudo-relevance feedback may hurt the search performance of this type of queries (Croft *et al.* 2010, p.283).

3. S-QL performs statistically significantly better than QL and AUX but slightly worse than ORG. This indicates that for this search task, the mixture-model based document smoothing approach of using anchor text for search performs a little more effectively than the semi-structured approach that combines query likelihood scores from different fields. One plausible reason is that both fields use language that has no huge difference, thus using structure information for this search task performs not as effectively as the approach of mixing information in different fields together. Nevertheless, both S-QL and ORG achieve statistical better performance than QL and AUX by using original anchor text information for search.

4. ORG-RALM performs statistically significantly better than ORG. The performance of 55 queries (30.4% of all 181 test queries) has been changed where ORG-RALM improves ORG on 39 queries. This indicates that the implicit anchor terms discovered by RALM provides additional information not in the original anchor text so that combining them can further improve the average retrieval performance.

5. ORG-RALM and RALM perform statistically significantly better than ORG-AUX and AUX, respectively. This indicates that, in the GOV2 collection, anchor text information discovered by the content based approach helps retrieval more effectively than the link based approach that uses auxiliary anchor text.

In Table 3.9, we also observe that the auxiliary anchor text helps the performance very little in this task. There are two plausible reasons: first, TREC NP queries are

short queries and Metzler *et al.*(2009) observed that auxiliary anchor text does not help or even hurts the performance of short navigational web queries; second, the anchor text sparsity problem is serious on the GOV2, thus a very small percentage of pages can collect auxiliary anchor text (as shown in Table 3.1) to benefit the search task. However, even when serious anchor text sparsity exists and queries are short, the content based approach still helps improve retrieval effectiveness. When comparing RALM's performance with results of 11 participants in TREC 2006 NP finding tasks (Büttcher *et al.* 2006), ORG-RALM's result can be ranked at $6th$ and beats all the runs that only use anchor text and page content for the task, except one that uses a complex machine learning approach and both unigram and bigram document features (Cen *et al.* 2006). We expect the content based approach can enhance the retrieval performance of general web search engines where there is a large portion of short navigational queries.

3.5 Conclusions

In this chapter, we employed our general perspective of discovering implicit information for search in Figure 1.1 (in Chapter 1) to address the anchor text sparsity problem in web search. We presented and compared webpage-side and query-side approaches of discovering implicit anchor text information for web search.

For the webpage-side approach (depicted in Figure 3.2(a)), we proposed a language modeling based method that uses web content similarity for discovering plausible anchor text. This content based method computes a relevant anchor text language model (called RALM) from a web page's similar pages' original anchor text for the anchor text discovery. Compared with a link based approach (Metzler *et al.* 2009), this content based approach has no specific link structure requirements on the web page of interest. We designed experiments with two TREC web corpora to evaluate the relative quality of the discovered anchor terms by three different

approaches: the link based approach, the RALM approach, and the keyword based approach. Experiments on the *simulated* web pages with no observed anchor text showed that the RALM approach can effectively discover hidden original anchor text and performed statistically significantly better than the two link based method on both collections.

For the query-side approach (depicted in Figure 3.2(b)), we presented how we adapt the Structured Relevance Model and use similar semi-structured records' information to discover plausible implicit query field values for search. The basic procedure is: add some structure to an unstructured query, view both queries and web pages as semi-structured records, build SRM based on the observed incomplete query fields, and then search web page records that are similar to the built SRM.

We evaluated retrieval performance of the two approaches (webpage-side and query-side) above and the link based approach (Metzler *et al.* 2009) with the TREC named page finding task. The results showed that for this search task, the webpage-side approach that discovers web pages' plausible anchor text and uses it to smooth document language model performed more effectively than the query-side approach that discovers implicit anchor text information in the query fields and uses extended query to retrieve similar records. Moreover, the content based approach helped retrieval more than the link based approach in this task; RALM can effectively discover information indicated by the observed original anchor text and further improved the retrieval performance. RALM can help improving retrieval effectiveness for short navigational queries even when serious anchor text sparsity exists. This makes RALM a promising technique for improving general web search engines. In future work, it is worthwhile to explore how well RALM can help long informational web queries.

CHAPTER 4

DISCOVERING MISSING CLICK-THROUGH INFORMATION IN QUERY LOGS FOR WEB SEARCH

4.1 Introduction

In this chapter, we address another problem that exists in a web search scenario where the web query log information is used to help improving retrieval performance of web search engines. We start with a detailed description of this research issue.

In a simplified web search scenario, a web user issues a query to a web search engine and obtains a ranked list of search results. Then the user reads the returned results, and may or may not click some of them to find his or her desired information. After that, the user may issue new queries to find more information on the same topic or start to explore new search topics. Typically, web search engines will record all the above interactions between web searchers and the engines in web query logs, which can be used later to improve the search engines' retrieval performance. Table 4.1 shows some example query log records in the Microsoft Live Search 2006 search query log excerpt (MS-QLOG)[1]. In this table, each row is a query log record that contains some important information about a user *click-through* event for a user-issued query. For each record, the second column (or field) shows the query content; the third column shows when the click event happened; the fourth column shows the URL of the clicked web page in this event; the fifth column shows the clicked URL's position on the URL rank list in the search result page returned by the search engine; the first column is a unique id for the click events that were triggered by a web searcher after he or she

[1] `http://blogs.msdn.com/livesearch/archive/2006/06/02/614486.aspx`

QueryID	Query	Time	URL	Pos.
dc3a05b1576a4a8a	newhaven register	2006-05-03 08:26:27	http://www.royahakakian.com /newsletter/NewHavenRegister.html	8
e13801b853444f56	newhaven register	2006-05-09 07:55:54	http://001forever.proboards42.com /index.cgi?action=register	1
8fbc6612038a466a	stories on mars	2006-05-18 16:08:43	http://www.steampunk.com /sfch/bibliographies/mars.html	7
5851612ed93b4733	raymond z. gallum	2006-05-20 03:08:46	http://www.steampunk.com /sfch/bibliographies/mars.html	5
5851612ed93b4733	raymond z. gallum	2006-05-20 03:09:37	http://zzmaster.best.vwh.net /SF/amazing_aa.html	4
fbd8e52e4ca64b3e	newhaven register classifieds	2006-05-22 05:39:03	http://www.royahakakian.com /newsletter/NewHavenRegister.html	3

Table 4.1. Some query log records from the Microsoft Live Search 2006 search query log excerpt.

issued the same query to the search engine. Note that typically the web queries that do not lead to any click-through events are also recorded in web query logs. In that case, in MS-QLOG both the clicked URL and its position field are empty and the time-stamp field records the time when the query was issued.

Click-through data provide very useful information for web search and play important roles in designing and improving web search engines. In one aspect, the user click-through information in web query logs reveals each web searcher's implicit preference information on the returned search results for each query. Thus, the click-through data have been used to derive labeled training data (i.e., preference order labels between web pages for a given query) for optimizing web ranking functions used by web search engines (Joachims 2002; Radlinski and Joachims 2007; Carterette and Jones 2007); the user clicks have also been directly used as relevance indicators of the clicked web pages to generate evaluation data for comparing the retrieval performance of different web search methods (Bendersky and Croft 2009). In another aspect, the click-through data also contain users' collective web search information that is very useful for extracting effective web page ranking features to enhance ranking models of web search engines (Xue *et al.* 2004; Burges

et al. 2005; Agichtein *et al.* 2006; Gao *et al.* 2009). For example, Gao et al. (2009) demonstrated that two pieces of click-through information of a web page – the number of queries leading to the clicks on the page and the number of unique words in these *click-associated* queries – are especially helpful to improve the retrieval effectiveness of an artificial-neural-network based web ranking model called LambdaRank (Burges *et al.* 2006).

However, click-through data usually suffer from a data sparseness problem where large volume of queries have few or no associated clicks (Craswell and Szummer 2007; Gao *et al.* 2009). This problem may be caused by two related user click behaviors. One is that users may only click a very limited number of pages for a query so that the clicks are not complete; the other one is that users may just browse the returned snippets to fetch some useful information while not clicking any results even they are relevant. Gao et al. (2009) referred to these two situations as the *incomplete click* problem and the *missing click* problem, respectively. The incomplete/missing click problems greatly limit the possibility and reliability of using click-through features for web search tasks. For example, with incomplete clicks, click-through features may not be reliably computed with the limited number of available query-page click pairs; with no clicks, click-through features cannot even be extracted. To overcome click-through sparseness, Craswell and Szummer (2007) first extracted queries and their clicked URLs in a query log to build a query-URL bipartite click graph and then proposed a random walk algorithm on the graph to find plausible clicks between queries and web image URLs. We use the queries and their clicked web page URLs in Table 4.1 to illustrate their approach in Figure 4.1. The queries/URLs that have the same content are assigned with the same query/URL node ID, as shown in Table 4.2. The built query-URL click graph is shown in Figure 4.1(a). Their approach utilized the rank walk algorithm to discover plausibly closely related query and URL nodes, which are not directly linked but reachable on the

Query Node ID	Query	URL Node ID	URL
q_1	newhaven register	d_1	http://www.royahakakian.com /newsletter/NewHavenRegister.html
q_1	newhaven register	d_2	http://001forever.proboards42.com /index.cgi?action=register
q_2	stories on mars	d_3	http://www.steampunk.com /sfch/bibliographies/mars.html
q_3	raymond z. gallum	d_3	http://www.steampunk.com /sfch/bibliographies/mars.html
q_3	raymond z. gallum	d_4	http://zzmaster.best.vwh.net /SF/amazing_aa.html
q_4	newhaven register classifieds	d_1	http://www.royahakakian.com /newsletter/NewHavenRegister.html

Table 4.2. The correspondence between the query/URL nodes in Figure 4.1 and the queries/URLs in Table 4.1

click graph. Using their approach, two plausible missing links (depicted as dashed lines in Figure 4.1(b)) are discovered for the original click graph in Figure 4.1(a). The intuition behind the random walk approach is that the semantic relation exists among different queries that led to the clicks on the same page and among different pages that are clicked due to the same user-issued query; thus, the transitions of the semantic relation on the click graph can be used to discover plausible clicks between queries and URLs.

The random walk approach can only partially alleviate the click-through sparseness problem because it requires specific link structures in the bipartite click graph to discover new clicks. For example, URLs (web pages) that have not yet received any clicks in the search history can never be associated with any previously issued queries in the query logs, even though the queries and the pages may have close semantic relation. Therefore, the expanded click-through features still suffer from the incomplete/missing click problems, where only a limited number of query-URL click pairs are available for feature extraction even after the bipartite click graph is enriched using the random walk algorithm. To address this issue, Gao et al. (2009) considered an alternative approach to compute click-through features from sparse click-through

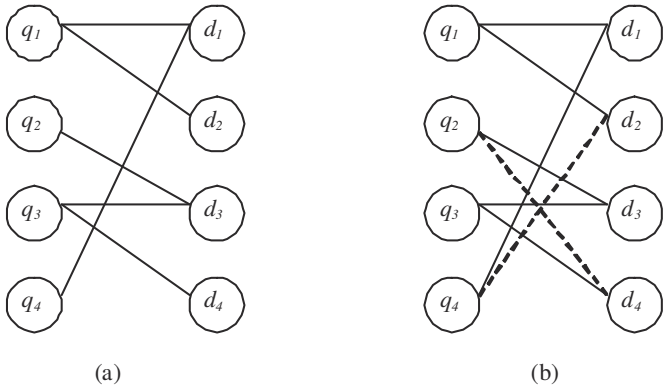


Figure 4.1. An illustration example of building a query-URL click graph from Table 4.1 and using random walk approach to discover plausible missing clicks: (a) the original built click graph; (b) the link-enriched click graph after applying rank walk algorithm on the original one.

data. They introduced a Good-Turing estimator (Good 1953) based discounting method to smooth click-through features of web pages, so that web pages that do not receive any click can have very small *non-zero* click-through features computed by discounting the average of the click-through features of all web pages that receive exactly *one* click. Intuitively, their approach follows the smoothing approach of computing out-of-vocabulary (OOV) words' probabilities in statistical language models to compute missing click-through features for web pages. They demonstrated that using smoothed click-through features to learn ranking models performed *statistically* significantly better than not doing smoothing on three web search datasets, including two large-scale Microsoft proprietary query logs, three web query sets and their human-labeled relevance judgment.

Notice that although OOV words in unseen documents and missing clicks of web pages can be both viewed as events unseen in training data and thus handled in a similar way, there is some important difference between two different unseen events. That is, we usually know little semantic information about the OOV word while we normally have already crawled the content of the web pages that have not received

clicks yet. However, Gao et al.'s approach described above does not use any semantic information in the web page content, thus pages that have completely *different* content but no clicks will obtain the *same* smoothed click-through feature value. This is counter-intuitive and makes many smoothed features less useful for ranking. Indeed, in their experiments, Gao et al. (2009) found that using the click-through features extracted from a web page's click-associated queries' content (called *query-dependent* features by them because these features depend on the content of the query strings), their smoothing approach helped little for improving retrieval performance; in contrast, using two smoothed click-through features (the number of click-associated queries of a page and the number of words in these queries) that contain little semantic information (called *query-independent* features by them) consistently and effectively improved retrieval performance in different web search tasks[2].

To overcome the weakness of both Gao et al.'s smoothing approach and Craswell and Szummer's random walk approach (2007), we propose to utilize the content similarity between web pages to address the click-through sparseness problem. Different from the Good-Turing estimator based smoothing approach, our content based approach is able to discover language modeling based click-through features that can properly convey semantic information in the web page content. Different from the random walk approach, we do not need the specific click graph structure to discover incomplete/missing clicks for web pages, thus can reduce the click-through sparseness further.

Our content based approach is shown in Figure 4.2(a) (also shown in Figure 1.5 in Chapter 1). We hypothesize that *web pages that are similar in content may be clicked by web searchers issuing similar queries* due to the semantic relation between queries and the web page content of their clicked URLs. Under this assumption, we develop

[2]One plausible reason is that these two features imply the popularity of each web page.

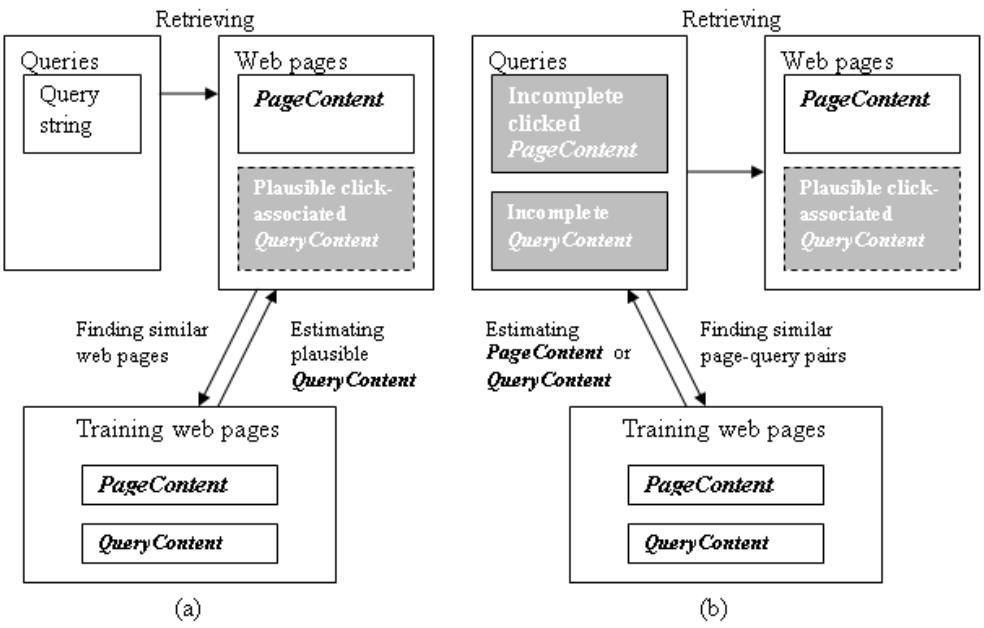


Figure 4.2. The specific perspective of discovering missing click-through features for web search: (a) using similar web pages for discovering additional click-associated queries; (b) finding similar page-query pairs to reconstruct better queries for search.

a language modeling based technique for discovering a target web page's plausible click-associated queries by using the queries that led to the clicks on pages similar to the target page. Then the discovered features are used for retrieval. Because we are also interested in the query-side discovery approach for addressing the click-through sparseness problem, we consider an approach depicted in Figure 4.2(b). This approach adds structure to an unstructured web query and attempts to directly discover the implicit information in the query fields by using the Structured Relevance Model (SRM) approach described in Chapter 2; then the expanded semi-structured query is used for retrieval.

Because we are particularly interested in different ways of using sparse click-through data for improving web search, we further consider an approach that combines the advantages of both the click graph based random walk approach and our proposed content based approach. In this approach, we first discover plausible links

in the click graph using the random walk approach, then employ our content based approach to discover click-through query language features for web pages using the enriched click graph.

We design evaluation experiments with the MS-QLOG dataset (briefly mentioned at the beginning of this chapter and further described in later sections) and two different sets of *ad hoc* web search tasks: (1) the ones in the TREC 2004-2005 Terabyte Tracks (Clarke *et al.* 2005; Clarke *et al.* 2004) and (2) the ones in the TREC 2009-2010 Web Tracks (Clarke *et al.* 2009; Koolen and Kamps 2010)[3]. Three retrieval approaches, including our content based approach, the query-side discovery approach and the approach of combining click graph and web content information for search, are evaluated.

The remaining parts of this chapter will be organized as follows. We begin by reviewing related work in §4.2. Next, in §4.3, we describe three webpage-side approaches of discovering missing click-through information for web pages with few or no clicks. After that, in §4.4 we present how to incorporate the discovered information into retrieval models for helping search. In §4.4.1, we present language modeling based retrieval models that utilize web pages' discovered click-associated query language models for improving search performance; then we formally describe the SRM based query-side approach of discovering implicit click-through features for search in §4.4.2. In §4.4.3 we design experiments to compare the retrieval performance of different approaches. We conclude in §4.5.

4.2 Related Work

Previous research has demonstrated the data sparseness problem in click-through data, including the incomplete click problem and the missing click problem, when

[3] `http://plg.uwaterloo.ca/~trecweb/2010.html`

leveraging web query logs for helping different web search tasks (Craswell and Szummer 2007; Agichtein *et al.* 2006; Radlinski and Joachims 2007; Xue *et al.* 2004; Li *et al.* 2008; Seo *et al.* 2011). However, there is relatively little work directly handling click-through sparseness for web search itself. As mentioned in §4.1, Craswell and Szummer (2007) proposed a random walk algorithm on the query-URL bipartite click graph to find plausible clicks; Gao et al. (2009) proposed a discounting method inspired by the Good-Turing estimator (Good 1953) to smooth click-through features for web pages that have received no clicks, in order to improve web search results. Gao et al. (2009) also considered combining Craswell and Szummer's random walk approach with their click-smoothing approach to achieve better retrieval performance. Here we also directly address the click-through sparseness problem. Different from previous work, we propose using web content similarity to discover click-through features for search. We also combine our content based approach with the random walk approach to further reduce click-through sparseness and improve retrieval performance. Radlinski et al. (2007) considered the missing click problem caused by a search engine's ranking bias and proposed an active learning approach to collect more click-through data by adjusting the search engine's returned rank list. Unlike their work, our approach computes plausible click-through features for web documents off-line and involves no human labeling efforts, thus can save online processing time. Recently, Seo et al. (2011) proposed applying spectral graph analysis on the web content similarity graph to smooth click counts in the web query logs and then using the smoothed counts for improving search. Our approach is similar to their approach in terms of using web content similarity to address click-through sparseness; however, we specifically focus on discovering missing semantic click-through features for helping search.

Our approach is related to other similarity based techniques, such as cluster-based smoothing from the language modeling framework (Kurland and Lee 2004;

Kurland and Lee 2006; Liu and Croft 2004; Tao *et al.* 2006), except we focus on enriching web pages' semantic click-through features for web search by using their similar pages' click-associated queries. We further consider combining web content similarity and click graph information to improve web search. We notice that Li et al. (2008) also considered combining web content and click graph information for mitigating the click-through sparseness they experienced when classifying web search intents of queries in web query logs.

As mentioned in §4.1, there is significant research work on using click-through data in the query log for enhancing web search performance. Some research considered using query-URL click-through pairs to derive labeled training pairs for learning web page ranking functions (Joachims 2002; Radlinski and Joachims 2007; Carterette and Jones 2007); other research focused on directly extracting click-through features and incorporating them into ranking models for web search (Xue *et al.* 2004; Burges *et al.* 2005; Agichtein *et al.* 2006; Gao *et al.* 2009). The incomplete/missing click problems present major challenges for both approaches of using click-through data for web search. Our research on discovering additional click-through features can benefit the latter research direction in particular.

Similar to our approach for discovering plausible anchor text for web pages in Chapter 3, we use a content-based, contextual translation approach to discover plausible click-through features for pages with no clicks from their similar pages' click-through features. Moreover, by viewing click-associated queries as a special semi-structured textual field of a web page and treating web queries as semi-structured short web pages, we adapt the Structured Relevance Model approach (Lavrenko *et al.* 2007), described in Chapter 2, for a query-side discovery for the click-through challenge addressed in this chapter.

4.3 Discovering Missing Click-through Information for Web Pages

We first describe two different approaches of discovering plausible click-through information for web pages with few or no clicks in web query logs. We then present one way to combine the two approaches to further reduce click-through sparseness. In our research, we are particularly interested in obtaining click-through features that can convey some semantic information of the target web page for search; therefore, we focus on discovering each web page's plausible (but missing) *click-associated* queries (i.e., queries that may lead to the clicks on the target page). We start with describing the random walk approach that uses co-click information in the click graph to discover plausible missing clicks (Craswell and Szummer 2007; Gao *et al.* 2009).

4.3.1 Finding More Click-associated Queries through Random Walk on Click Graph

In the introduction (§4.1), we described how to build a query-URL bipartite click graph from a web query log and briefly introduced the procedure of employing the random walk approach to discover plausible clicks between query nodes and URL nodes. Intuitively, the random walk approach assumes that there exists some semantic relation among different queries that led to the clicks on the same page and among different pages that are clicked due to the same user-issued query. This assumption can be used for discovering new plausible clicks between queries and URLs.

Formally, assume that the bipartite click graph $G =< Q, U, E >$ is constructed from a set of query nodes $Q = \{q_1 \ldots q_m\}$, a set of web page URL nodes $U = \{u_1 \ldots u_n\}$ and the edges E between the query nodes and the URL nodes. $(q_i, u_j) \in E$ is an edge in G when q_i leads to at least one click on u_j, and $w(q_i, u_j)$ represents the click count associated with the edge (q_i, u_j). We can normalize the $w(q_i, u_j)$ to obtain the

transition probability $p(u_j|q_i)$ on the click graph between a query q_i and each of its clicked web page u_j by:

$$p(u_j|q_i) = \frac{w(q_i, u_j)}{\sum\limits_{k\in\{1...n\},(q_i,u_k)\in E} w(q_i, u_k)}, \tag{4.1}$$

and also the transition probability $p(q_i|u_j)$ between a page u_j and each of its click-associated queries q_i by:

$$p(q_i|u_j) = \frac{w(q_i, u_j)}{\sum\limits_{k\in\{1...m\},(q_k,u_j)\in E} w(q_k, u_j)}. \tag{4.2}$$

We can use the above transition probabilities $p(u_j|q_i), p(q_i|u_j), i \in \{1...m\}, j \in \{1...n\}$ to compute the probability $p^{(2t)}(q_j|q_i)$ of one query, q_i, transitioning to another, q_j, on the click graph in $2t$ steps by the following iterative equations:

$$\begin{aligned}
p^{(2t)}(q_j|q_i) &= \sum_{k\in\{1...n\},(q_j,u_k)\in E} [p(q_j|u_k)p^{(2t-1)}(u_k|q_i)], t \geq 1; \\
p^{(2t-1)}(u_j|q_i) &= \sum_{k\in\{1...m\},(q_k,u_j)\in E} [p(u_j|q_k)p^{(2t-2)}(q_k|q_i)], t > 1; \\
p^{(1)}(u_j|q_i) &= p(u_j|q_i), i \in \{1...m\}, j \in \{1...n\}.
\end{aligned} \tag{4.3}$$

We can see that longer transition steps can discover transitions to additional queries for a target query q_i while the discovered semantic relation between them becomes weaker and noisier. Thus for effectiveness and efficiency, we follow Gao et al.(2009) to set $t = 1$ in our experiments. In order to reduce noise, we follow their approach and require that the discovered transitions for the target query q_i should satisfy $p^{(2)}(q_j|q_i) > \alpha$, where α is a controlling parameter and tuned empirically on training data for different tasks[4].

[4]In Gao et al.(2009)'s original experiments, they only kept up to 8 similar queries that satisfy $p^{(2)}(q_j|q_i) > \alpha$ for each query q_i for efficiency. We do not apply this additional restriction, because the MS-QLOG dataset contains much less queries/URLs click pairs than their data.

After discovering related queries for each query using the random walk approach, Gao et al.(2009) expanded each web page's click-associated queries with the discovered related queries. In this way, they can link web pages with more queries that may be semantically related to the content of the pages so that the incomplete click problem is partially mitigated. Then they used the enriched representation of the click-associated queries of each page to extract useful click-through features to improve web search performance. Table 4.3 shows some summary statistics of the original query-URL bipartite click graph and the enriched click graphs by the random walk approach when we use the click pairs in the MS-QLOG dataset to build the click graph. The first four rows in Table 4.3 show some summary statistics of the original click graph built from MS-QLOG, indicating that the click-through information is very sparse even for the clicked pages [5] – on average, each web page only received 2.5 clicks and has about 1.4 unique click-associated queries, and each query only leads to about 3.5 clicks. The last five rows show the number of click edges in each enriched graph by the random walk approach using different noise filtering parameter values, indicating that incomplete click problem can be partially mitigated: on average, the number of the unique click-associated queries of each web page has been raised to 6.5 when $\alpha = 0.001$, and 3.2 when $\alpha = 0.01$ and 8 most similar queries were used (as in Gao et al.'s experiments (2009)), respectively.

4.3.2 Discovering Missing Click-associated Queries through Finding Similar Pages

Notice that the random walk approach needs specific click graph structure to discover plausible missing clicks: it cannot handle web pages with no clicks. Therefore, we propose to use our content based approach to discover plausible click-associated

[5]MS-QLOG does not contain the URLs that were not clicked by the users; thus we have no information about the pages with no clicks from MS-QLOG.

# click pairs	#unique queries (query nodes)	#unique URLs (URL nodes)
12,251,067	3,545,174	4,971,990
	# unique click edges in the graph	
original counts	6,853,498	
enriched by random walk($t = 1$)		
$\alpha = 0$ (no noise filtering)	42,999,932	
$\alpha = 0.001$	32,240,647	
$\alpha = 0.005$	24,365,787	
$\alpha = 0.01$	20,265,365	
$\alpha = 0.01$ and 8 most similar queries	16,041,102	

Table 4.3. Some summary statistics about the original click graph built from the click events in the MS-QLOG dataset and the edge counts of the enriched graphs by the random walk approach with different noise filtering parameters.

queries for a web page. Intuitively, our approach assumes that web pages that are similar in content may receive clicks from web searchers issuing similar queries (due to the semantic relation among similar pages as well as pages and their click-associated queries). Under this assumption, we aim to discover a query language model for each page, in order to obtain effective missing semantic click-through features to help search.

Our approach adapts the content based approach of discovering anchor text (§3.3.2) to handle missing click-through query information here. In the anchor text discovery task there, we first view the content of web pages as their anchor text's descriptive context and utilize the contextual translation approach (Wang and Zhai 2008) to measure the semantic relation between the anchor text associated with different pages. Given any page P_i and a target page P_0, the semantic relation between their associated anchor text A_i and A_0 is measured by the contextual translation probability $t(A_i|A_0)$, computed from the Kullback-Leibler divergence (KL-div) between the document language models of P_i and P_0. Then we can use $t(A_i|A_0)$ to compute a relevant anchor text language model $p(w|A_0)$ for a target page P_0 to discover P_0's plausible implicit anchor terms by:

$$p(w|A_0) = \sum_{A_i \in \mathcal{A}} p(w|A_i) \times t(A_i|A_0), \tag{4.4}$$

where $\mathcal{A}$ denotes the complete anchor text space of all pages and $p(w|A_i)$ is a multinomial distribution of anchor terms (w) over the vocabulary $\mathcal{V}_{\mathcal{A}}$.

Similarly, here we first view each page P_i's content as the descriptive context of the page's click-associated queries Q_i and use P_i's document language model, $p_i = \{p(w|P_i)\}$, as Q_i's contextual language model, which is also computed by applying Dirichlet smoothing on the original un-smoothed document language model:

$$p(w|P_i) = \frac{N_{P_i}}{N_{P_i} + \mu} p_{ML}(w|P_i) + \frac{\mu}{N_{P_i} + \mu} p(w|\mathcal{C}), \tag{4.5}$$

where $p_{ML}(w|P_i)$ is the maximum likelihood (ML) estimate of observing a word w in the page, $p(w|\mathcal{C})$ is w's probability in the collection $\mathcal{C}$, N_{P_i} is the length of P_i's content and μ is the Dirichlet smoothing parameter.

Then given any page P_i and a target page P_0, we measure the semantic relation between their click associated queries Q_i and Q_0 by their contextual translation probability $t(Q_i|Q_0)$, computed from the KL-div $Div(\cdot||\cdot)$ between their contextual models p_0 and p_i:

$$t(Q_i|Q_0) = \frac{\exp(-Div(p_0||p_i))}{\sum_i \exp(-Div(p_0||p_i))} \propto \prod_w p(w|P_i)^{p(w|P_0)}. \tag{4.6}$$

The end of Equation 4.6 is the likelihood of generating Q_0's context P_0 from the smoothed language model of Q_i's context P_i, being normalized by Q_0's context length.

After that, for each given target page P_0, we calculate a relevant (click-associated) query language model (**RQLM**) $p(w|Q_0)$ to discover P_0's plausible click-associated query terms by:

$$p(w|Q_0) = \sum_{Q_i \in \mathcal{Q}} p(w|Q_i) \times t(Q_i|Q_0), \tag{4.7}$$

where Q_i denotes all the queries that may lead to the clicks on P_i but may be *incomplete* or *missing*, $\mathcal{Q}$ denotes the complete textual space of the click-associated

queries of all pages, $p(w|Q_i)$ is a multinomial distribution of query terms (w) over the click-associated query language vocabulary $\mathcal{V}_\mathcal{Q}$.

To compute the RQLM $p(w|Q_0)$ in Equation 4.7, we use each page P_i's click-associated queries originally *observed* in the query log to estimate a query language model $p_{obs}(w|Q_i)$ to approximate $p(w|Q_i)$, which would ideally be estimated from some unknown *complete* set of P_i's all plausible click-associated queries in the query log[6]. In practice, for effectiveness and efficiency we compute the RQLM of the target page P_0 using the click-associated queries of P_0's top-k most similar pages in the query log. This choice is due to two reasons: (1) $t(Q_i|Q_0)$ is very small for other pages thus has less impact on the RQLM; (2) increasing k can increase the number of query samples for better estimating RQLM but also may introduce more noise to degrade the quality of the estimated RQLM. We tune k's value on the training data for each different retrieval task.

4.3.3 Combining Random Walk Approach and Finding Similar Approach

We can use both the random walk approach (in §4.3.1) and our content based approach (in §4.3.2) to further reduce the click-through sparseness and obtain better semantic click-through features for search. Here we present one language modeling based way to combine the advantages of two approaches.

We first employ the random walk approach to enrich the original bipartite click graph and discover more click-associated queries for each page. Then we estimate a query language model $p(w|Q_{aug})$ for each web page from the new added click-associated queries, which we call *augmented* queries, of the page. We also estimate a query language model $p(w|Q_{orig})$ for each page from the click-associated queries originally observed in the query log, which has *not* been enriched by the random

[6]We will use this fact in §4.3.3 to combine the random walk approach and the content based approach for discovering missing click-through features.

walk approach. Next, we employ the mixture model approach (Nallapati *et al.* 2003; Ogilvie and Callan 2003) to combine two query language models $p(w|Q_{orig})$ and $p(w|Q_{aug})$, and compute a better smoothed query language model $\tilde{p}(w|Q)$ by:

$$\tilde{p}(w|Q) = \gamma p(w|Q_{orig}) + (1-\gamma)p(w|Q_{aug}), \tag{4.8}$$

where γ is a meta-parameter to control the mixture weight (or prior probability) of each component and be tuned on training data for different tasks. Then we use the updated query language model $\tilde{p}(w|Q)$ of each page to better approximate the $p(w|Q_i)$ in Equation 4.7 so that we can better estimate the RQLM $p(w|Q_0)$ of each page P_0 to help retrieval.

We now describe how we use discovered click-through features to help search.

4.4 Using Information Discovered from Query Logs for Web Search

Similar to how we leverage different discovered anchor text information for retrieval (in §3.4), we consider two alternative retrieval approaches shown in Figure 4.2(a) and (b) (in the introduction of this chapter). The webpage-side approach (in Figure 4.2(a)) utilizes discovered semantic click-through features of web pages for re-ranking them. The query-side approach (in Figure 4.2(b)) constructs semi-structured records to use semantic click-through features and then employs the Structured Relevance Models approach (in §2.5.1) for search. For the convenience of discussing different retrieval models and baselines, we start by briefly describing the data and methodology we used for evaluating different approaches.

Mainly due to privacy and security concerns, there are very limited publicly available web query log data even for academic research purpose. In our experiments, we use the Microsoft Live Search 2006 search query log excerpt (MS-QLOG), which

has been used in some previous query log study (Bendersky and Croft 2008b; Wang and Zhai 2008; Bendersky and Croft 2009). We have briefly described this dataset in the introduction of this chapter. MS-QLOG contains click-through information of 12,251,068 click-through events and also information of 14,921,286 additional user-issued web queries that received no clicks, both sampled from the query log of Microsoft's web search engine during 05/01/2006 to 05/31/2006. We only use the click-through records in this dataset for our experiments.

For our retrieval experiments, we use the queries and the corresponding *human-labeled* relevance judgments in two TREC web search tasks. The first one consists of the *ad hoc* web search tasks in the TREC 2004-2005 Terabyte Tracks (Clarke *et al.* 2005; Clarke *et al.* 2004) and the second one consists of the *ad hoc* web search tasks in the TREC 2009 Web Track (Clarke *et al.* 2009; Koolen and Kamps 2010) and the TREC 2010 Web Track[7]. The search was performed on the GOV2 collection (a standard TREC web collection crawled from government web sites during early 2004) in the first retrieval task, and on the category B subset of the ClueWeb09 Dataset[8] (another standard TREC web collection recently crawled from the Web during 01/06/2009 to 02/27/2009) in the second retrieval task, respectively. These are the same corpora but different tasks used in Chapter 3.

Because our approach depends on web page content similarity, we crawl the web content of all the clicked URLs in the MS-QLOG dataset and use the crawled pages and their click-associated queries in MS-QLOG as the training web pages depicted in the bottom external boxes of Figure 4.2(a) and (b) (in the introduction of this chapter). The GOV2 collection and the TREC category B subset of the ClueWeb09 web collection, known as the ClueWeb09-T09B dataset, are used as the searched target web collections depicted in the upper-right external boxes of Figure 4.2(a) and

[7] `http://plg.uwaterloo.ca/~trecweb/2010.html`

[8] `http://boston.lti.cs.cmu.edu/Data/clueweb09/`

(b). Each ClueWeb09 page or GOV2 page can be viewed as a page with no click information[9] thus both the training web pages and the searched items encounter the click-through sparseness problem. More details about the data and methodology used for evaluating the retrieval performance of different approaches will be described later (in §4.4.3).

Next, we describe our retrieval models, then discuss the experimental results in §4.4.3.

4.4.1 Document Smoothing Based Retrieval Models

The first baseline is the same as the query likelihood baseline described in §3.4.1 (which describes retrieval models using anchor text information). This baseline does not use any click-through features and ranks each web page P for a query Q by the likelihood of the page P's document language model $p(w|P)$ generating the query Q:

$$p(Q|P) = \prod_{w \in Q} p(w|P). \tag{4.9}$$

Again we use Dirichlet smoothing to compute the document language model $p(w|P)$ used in the above equation and denote this query likelihood baseline **QL** here. We tune the Dirichlet parameter μ in Equation 4.5 for QL to achieve the best retrieval performance for different tasks. Note that μ is fixed to 2500 when computing the document models of the crawled clicked URLs in MS-QLOG for estimating RQLMs (relevant click-associated query language models described in §4.3.2) for different tasks.

We also follow the mixture model approach (Nallapati *et al.* 2003; Ogilvie and Callan 2003) to use the discovered click-through query language model features to help search. After we estimate the RQLM $p(w|Q_0)$ for each page, we mix a web page

[9]Some previous research showed that there is very small overlap between the clicked URLs in MS-QLOG and the GOV2 collection (Bendersky and Croft 2009).

P's document language model $p(w|P)$ with the RQLM to obtain a better document language model $\tilde{p}(w|P)$ by:

$$\tilde{p}(w|P) = \beta p(w|P) + (1 - \beta)p(w|Q_0), \tag{4.10}$$

where $p(w|P)$ is the original smoothed document model in the QL baseline and β is the meta-parameter controlling the mixture weights of the component distributions. Then we use the updated document language model $\tilde{p}(w|P)$ to replace $p(w|P)$ in Equation 4.9 for retrieval.

We have described three different approaches of discovering plausible missing click-through features in §4.3. In our experiments, because the searched items are the ClueWeb09 or GOV2 web pages with no click information, only using the random walk approach cannot discover any click-associated queries for them to help search. Therefore, we only consider using our content based approach and the combination approach (in §4.3.3) for retrieval. In the combination approach, we first discover plausible links in the click graph of the MS-QLOG dataset by the random walk approach and then use the enriched click graph to estimate better RQLMs for the ClueWeb09 or GOV2 pages by our content based approach. We denote the retrieval baseline that uses RQLMs from the content based approach to update document models for search as **RQLM**, and the baseline that uses RQLMs from the combination approach for search as **RW+RQLM** in later discussions.

4.4.2 Query-side Implicit information Discovery for Search

Similar to the query-side approach of discovering anchor text for search, we employ a semi-structured query-side approach in Figure 4.2(b) (in the introduction of this chapter) to address the missing/incomplete click problem. We build a semi-structured query from each query in the *ad hoc* web search tasks and then utilize the Structured

Relevance Models (SRM) based retrieval approach to discover implicit query field values for retrieval.

Formally, we view each web page as a semi-structured record containing two fields: (1) the *Page Content* field (denoted by $\mathbf{w}_p$) which contains the original page content and (2) the *Query Content* field (denoted by $\mathbf{w}_q$) which contains all the click-associated queries of the page in the web query log. Then for each unstructured query q, we generate a semi-structured query $\mathbf{q} = \{\mathbf{w}_p, \mathbf{w}_q\}$ that has the same semi-structure as the web page record by duplicating the query string in both fields, i.e. $\mathbf{w}_p = \mathbf{w}_q = q$. We assume that both fields are incomplete and then use the SRM approach to estimate plausible implicit field values in $\mathbf{q}$ based on the observed $\{\mathbf{w}_p, \mathbf{w}_q\}$. We use our crawled pages of the clicked URLs in MS-QLOG and their click-associated queries in MS-QLOG to form the training semi-structured record collection $\mathcal{W}$.

We then use the same procedure described in §3.4.2 (which discussed the query-side retrieval model of discovering anchor text for search) and the training collection to calculate the SRM $\{R_p(\cdot), R_q(\cdot)\}$ for $\mathbf{q}$, where each relevance model $R_i(w)$ specifies how plausible it is the word w would occur in the field i ($i \in \{p, q\}$) of $\mathbf{q}$ given the observed $\mathbf{q} = \{\mathbf{w}_p, \mathbf{w}_q\}$.

In the process of computing the SRM for $\mathbf{q}$, we use the following equation to compute the posterior probability $P(\mathbf{w}'|\mathbf{q})$ of generating $\mathbf{q}$ from the training web page records $\mathbf{w}' \in \mathcal{W}$:

$$\begin{aligned} P(\mathbf{w}'|\mathbf{q}) &\propto P(\mathbf{q}|\mathbf{w}') * P(\mathbf{w}'), \\ P(\mathbf{q}|\mathbf{w}') &= P(\mathbf{w}_p|\mathbf{w}'_p)^{\beta_p} * P(\mathbf{w}_q|\mathbf{w}'_q)^{\beta_q}, \end{aligned} \tag{4.11}$$

where the meta-parameters β_p and β_q are used to control the impact of each field on the posterior probability and tuned with the training queries. In addition, when computing $P(\mathbf{w}_i|\mathbf{w}'_i), i \in \{p, q\}$ in Equation 4.11, we perform smoothing in each field

and fix the Dirichlet smoothing parameters $\mu_p = 50, \mu_q = 1$ for the *Page Content* and *Query Content* fields, respectively. [10]

Again, for efficiency and effectiveness we use $\mathbf{q}$'s top-k most similar records instead of all records $\mathbf{w}' \in \mathcal{W}$ to calculate $R_i(w)$. We tune the value of k with the training queries. Because the click information is completely missing in each of our two searched target collections $\mathcal{W}''$ (ClueWeb09-T09B and GOV2), the *Query Content* field is empty there. Therefore, we only use the relevance model $R_p(w)$ of the estimated SRM in the *Page Content* field to search each target collection. We interpolate it with the original query language model to obtain a better relevance model for retrieval:

$$R'_p(w) = \lambda * (p(w|\mathbf{w}_p)) + (1 - \lambda) * R_p(w), \tag{4.12}$$

where λ is used to control the impact of the original query language model on the updated relevance model and tuned with the training queries. Then the searched web page records $\mathbf{w}'' \in \mathcal{W}''$ are ranked by their similarity to $R'_p(w)$:

$$H(R'_p; \mathbf{w}''_p) = \sum_{w \in \mathcal{V}_p} R'_p(w) \log p(w|\mathbf{w}''_p). \tag{4.13}$$

We denote this retrieval baseline as **SRM** in the experiments.

4.4.3 IR Experiments

4.4.3.1 Data and Methodology

We have described the GOV2 collection, the ClueWeb09 collection and its TREC category B subset (ClueWeb09-T09B) earlier and also in §3.3.4.1 where we designed experiments with these collections to examine the quality of the discovered anchor

[10] When we used some sampled queries in the MS-QLOG to search their clicked URLs in our crawled training web collection, we found that using these smoothing parameters can achieve the best retrieval performance, if the user click is directly used as the relevance indicator of the web page. Note that this way can only obtain very sparse, biased and incomplete relevance judgments, so we do not use it for designing retrieval experiments for evaluation.

terms by different approaches. Here we use GOV2 and ClueWeb09-T09B as the searched target collection in the first and second retrieval task, respectively. We use the Indri Search Engine[11] to index each collection by removing a standard list of 418 INQUERY (Broglio *et al.* 1993) stopwords and applying the Krovetz stemmer.

For the first retrieval task, we use 50 *ad hoc* query topics (TREC topic id:701-750,title-only) and their relevance judgments in the TREC 2004 Terabyte Track (Clarke *et al.* 2004) for training and 50 *ad hoc* query topics (TREC topic id:751-800,title-only) and their relevance judgments in the TREC 2005 Terabyte Tracks (Clarke *et al.* 2005) for testing. On average, there are about 210 judged relevant web pages per query in this retrieval task. For the second retrieval task, we use 50 *ad hoc* query topics (title-only) and their relevance judgments in the TREC 2009 Web Track (Clarke *et al.* 2009; Koolen and Kamps 2010) for training and the query topics (title-only) in the TREC 2010 Web Track for testing. On average, there are about 72 judged relevant web pages per query in this retrieval task. Moreover, the original *ad hoc* web search task in the TREC 2010 Web Track was performed on the whole ClueWeb09 collection; in contrast, here our searched target collection is ClueWeb09-T09B, a subset of ClueWeb09, thus we only use the human-labeled relevant pages in ClueWeb09-T09B for evaluation.

We crawled the web pages of the clicked URLs in the MS-QLOG during June 2010. We use these pages and their click-associated queries in the MS-QLOG as the training data for our experiments. Originally there are 4,971,990 unique clicked URLs in this query log, as shown in Table 4.3; we successfully crawled 3,031,348 HTML pages of the clicked URLs and indexed them using the Indri Search Engine. After removing 418 INQUERY stopwords and applying Krovetz stemmer, the indexed collection, which we call MS-QLOG-Web, contains about 21.5 million unique words

[11] `http://www.lemurproject.org/indri/`

and 4.1 billion word postings. These pages are then used for discovering missing click-through features for the GOV2 or ClueWeb09 pages. We also preprocessed the queries in the MS-QLOG using the same set of stopwords and stemming procedure.

To evaluate the retrieval performance of the different approaches, we calculate typical IR evaluation measurements including Mean Average Precision (MAP) and Precision at the top k-th rank position (P@k), which have been used in the IR experiments in the previous chapters. We also compute some IR measurements that use the graded relevance judgment information, including the Normalized Discounted Cumulative Gain (NDCG) and NDCG at position k (NDCG@k) (Järvelin and Kekäläinen 2002; Vassilvitskii and Brill 2006). For the TREC 2010 Web Track queries, the relevance score is an integer between [-2,3] with the most relevant page having score 3 and the most irrelevant page having score -2, because the TREC community began to provide a 6-level scale relevance judgment[12] for each query; for other query sets, relevance score is an integer between [0,2] with the most relevant page getting score 2. For the performance on the TREC 2009 Web Track queries, we also report two additional measurements: *statMAP* and *MPC(30)*, which were used by the TREC community for that track (Clarke *et al.* 2009) and computed by the evaluation tool *statAP_MQ_eval_v3.pl* [13] provided by the TREC community; thus, we can compare our results with other researchers' published results on the same query set. Intuitively, both statMAP and MPC(30) measurements are used for addressing the incomplete judgment issue (i.e. there may exist some relevant pages that have not got the chance to be judged; treating them as non-relevant pages may underestimate the actual IR performance): statMAP is a statistical version of the MAP measurement and MPC(30) is a statistical version of the measurement P@30 (Aslam and Pavlu 2007; Carterette *et al.* 2006).

[12] `http://plg.uwaterloo.ca/~trecweb/2010.html`

[13] It is downloadable at: `http://trec.nist.gov/data/web09.html`

In each retrieval task, we first tune the Dirichlet smoothing parameter μ in Equation 4.5 to obtain the best QL baseline that can achieve the highest MAP with training queries on each searched target collection (GOV2 or ClueWeb09-T09B). Then for both the RQLM baseline (using our content based approach) and the RW+RQLM baseline (using the combination approach), we employ the *reranking* approach, where we use the updated document language model by each approach to recompute the query likelihood scores of the top-1000 web pages returned by the QL baseline for each query and then rerank the pages. For the RQLM baseline, we tune these two parameters: the number (k) of the top-k similar pages whose click-associated queries are used to compute the RQLM and the mixture weight β in Equation 4.10. For the RW+RQLM baseline, we tune two additional parameters: the transition probability threshold α (discussed in §4.3.1) and the query language model updating weight γ in Equation 4.8. For the SRM baseline, as described in §4.4.2, we tune the number of the similar pages (k) used to build SRM, the number of terms (N) in each field of the built SRM for retrieval, the meta-parameters λ in Equation 4.12 and β_p, β_q in Equation 4.11. In each retrieval task, we tune the parameters of different approaches with the training queries, and then test the performance of different approaches with the tuned parameters on the test queries.

4.4.3.2 Results

Table 4.4 and 4.5 show the retrieval performance of different approaches with the training and testing queries, respectively, in the first retrieval task. Table 4.6 and 4.7 show the retrieval performance of different approaches with the training and testing queries, respectively, in the second retrieval task. Table 4.4 and 4.6 also show the corresponding tuned parameters of each approach in the first and second retrieval task, respectively. In addition, Figure 4.3 and 4.4 show the difference of the average precisions (AP) between RW+RQLM and QL, as well as between RW+RQLM and SRM,

	MAP	P@10	P@30	NDCG@1	NDCG	Optimal Param.
QL	0.2617	0.5102	0.4694	0.3741	0.4829	$\mu = 1000$
Query-side missing click-through feature discovery:						
SRM	$0.2777^{\ddagger}$	$0.5551^{\ddagger}$	$0.5020^{\ddagger}$	0.4150	0.4945	$k = 10, N = 50$ $\lambda = 0.3, \beta_p = 0.99$ $\beta_q = 0.01$
Webpage-side missing click-through feature discovery:						
RQLM	0.2688	$0.5388^{\ddagger}$	0.4796	$0.4422^{\dagger}$	$0.4927^{\ddagger}$	$k = 100, \beta = 0.95$
RW+RQLM	$0.2691^{\dagger}$	$0.5347^{\ddagger}$	$0.4823^{\dagger}$	$0.4490^{\ddagger}$	$0.4933^{\ddagger}$	$k = 100, \beta = 0.95$ $\alpha = 0.01, \gamma = 0.6$

Table 4.4. Retrieval performance and tuned parameters of different approaches on TREC 2004 Terabyte Track *ad hoc* queries (the training queries). The ‡ and † indicate statistically significant improvement over of the QL baseline based on one-sided t-test with $p < 0.05$ and $p < 0.1$,respectively.

	MAP	P@10	P@30	NDCG@1	NDCG
QL	0.3043	0.5560	0.4980	0.4667	0.5475
Query-side missing click-through feature discovery:					
SRM	0.3110	0.5700	0.5060	0.4400	0.5502
Webpage-side missing click-through feature discovery:					
RQLM	$0.3161^{\ddagger}$	$0.5960^{\ddagger}$	0.5120	$0.5133^{\dagger}$	$0.5601^{\ddagger}$
RW+RQLM	$0.3132^{\dagger}$	$0.5840^{\ddagger}$	0.5067	0.4800	$0.5579^{\ddagger}$

Table 4.5. Retrieval performance of different approaches on TREC 2005 Terabyte Track *ad hoc* queries (the test queries). The ‡ and † indicate statistically significant improvement over of the QL baseline based on one-sided t-test with $p < 0.05$ and $p < 0.1$,respectively.

	MAP	P@10	P@30	statMAP	MPC(30)	Optimal Param.
QL	0.1951	0.3408	0.3354	0.1732	0.3636	$\mu = 1000$
Query-side missing click-through feature discovery:						
SRM	$0.2258^{\ddagger}$	$0.4388^{\ddagger}$	$0.3959^{\ddagger}$	0.2069	0.4661	$k = 5, N = 100$ $\lambda = 0.4, \beta_p = 0.01$ $\beta_q = 0.99$
Webpage-side missing click-through feature discovery:						
RQLM	$0.2107^{\ddagger}$	$0.3714^{\dagger}$	$0.3694^{\ddagger}$	0.1916	0.4215	$k = 25, \beta = 0.9$
RW+RQLM	$0.2123^{\ddagger}$	$0.3796^{\ddagger}$	$0.3728^{\ddagger}$	0.1908	0.4359	$k = 25, \beta = 0.9$ $\alpha = 0.01, \gamma = 0.5$

Table 4.6. Retrieval performance and tuned parameters of different approaches on TREC 2009 Web Track queries (the training queries). The ‡ and † indicate statistically significant improvement over of the QL baseline based on one-sided t-test with $p < 0.05$ and $p < 0.1$,respectively.

	MAP	P@10	P@30	NDCG@1	NDCG
QL	0.1761	0.2292	0.2451	0.0714	0.3347
Query-side missing click-through feature discovery:					
SRM	0.1808	0.2354	0.2556	0.0595	0.3264
Webpage-side missing click-through feature discovery:					
RQLM	$0.1925^{\ddagger}$	$0.2646^{\dagger}$	$0.2708^{\dagger}$	$0.1339^{\ddagger}$	$0.3509^{\ddagger}$
RW+RQLM	$0.1995^{\ddagger}$	$0.2688^{\ddagger}$	$0.2715^{\dagger}$	$0.1339^{\ddagger}$	$0.3526^{\ddagger}$

Table 4.7. Retrieval performance of different approaches on TREC 2010 Web Track queries (the test queries). The ‡ and † indicate statistically significant improvement over of the QL baseline based on one-sided t-test with $p < 0.05$ and $p < 0.1$,respectively.

	statMAP	MPC(30)
QL	0.1442	0.3079
Anchor	0.0567	0.5558
Mix	0.1643	0.4812
UDWAxQEWeb	0.1999	0.5010
uogTrdphCEwP	0.2072	0.4966
ICTNETADRun4	0.1746	0.4368

Table 4.8. Retrieval performance of some published results on TREC 2009 Web Track *ad hoc* queries from other TREC participants. Results in 2nd-4th rows appeared in published work on using anchor text for web search. Results in 5th-7th rows are top3 best official submissions for this search task in the TREC 2009 Web Track among the TREC participants.

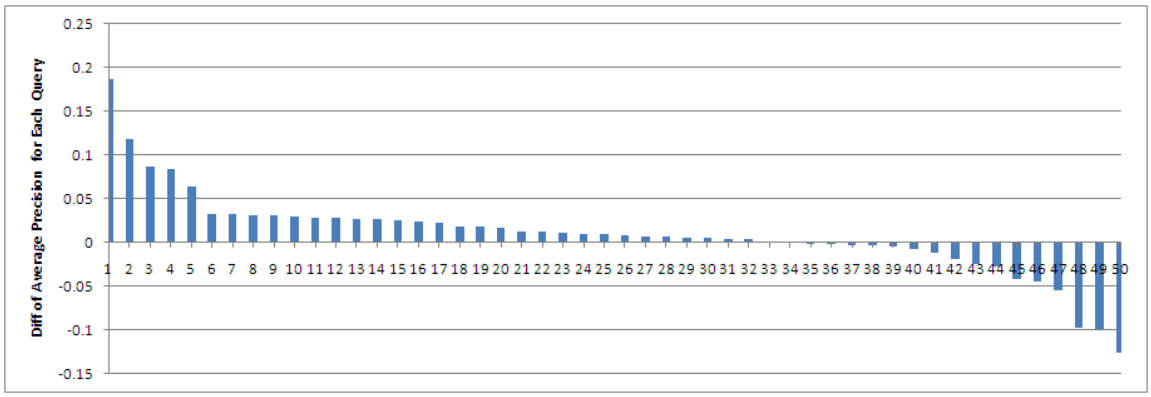


Figure 4.3. The difference of the average precisions (AP) between RW+RQLM and QL on each individual test query from TREC 2005 Terabyte Track. Above the x-axis reflect queries where RW+RQLM out-performs QL. Y-axis denotes the difference, computed using (RW+RQLM's AP minus QL's AP) of each test query. All the differences are sorted then depicted to show the IR performance difference of two approaches. Among 50 queries, RW+RQLM outperforms QL on 34 queries and performs worse than QL on 16 queries.

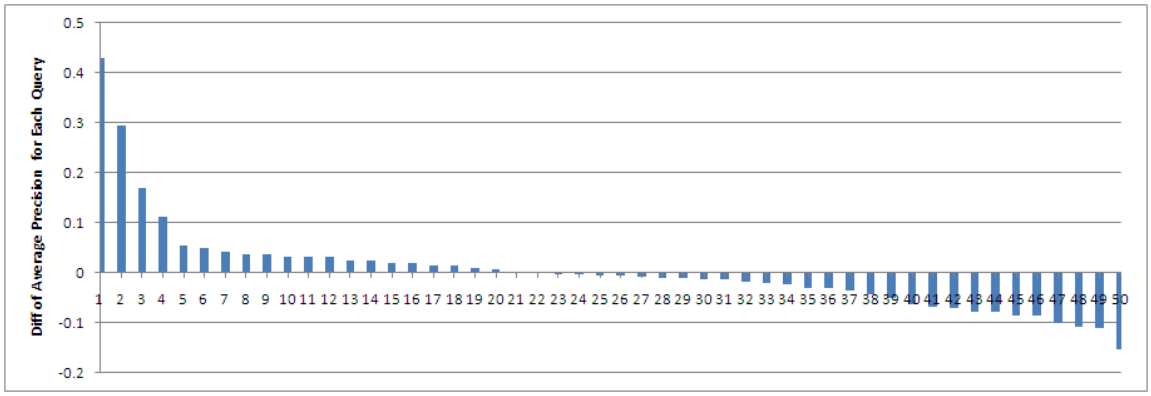


Figure 4.4. The difference of the average precisions (AP) between RW+RQLM and SRM on each individual test query from TREC 2005 Terabyte Track. Above the x-axis reflect queries where RW+RQLM out-performs SRM. Y-axis denotes the difference, computed using (RW+RQLM's AP minus SRM's AP) of each test query. All the differences are sorted then depicted to show the IR performance difference of two approaches. Among 50 queries, RW+RQLM outperforms SRM on 22 queries and performs worse than SRM on 28 queries.

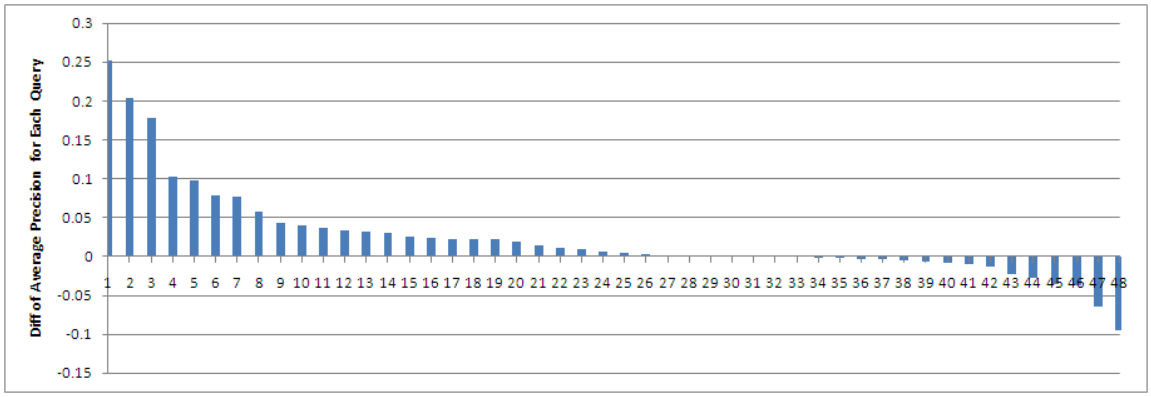


Figure 4.5. The difference of the average precisions (AP) between RW+RQLM and QL on each individual test query from TREC 2010 Web Track. Above the x-axis reflect queries where RW+RQLM out-performs QL. Y-axis denotes the difference, computed using (RW+RQLM's AP minus QL's AP) of each test query. All the differences are sorted then depicted to show the IR performance difference of two approaches. Among 48 queries (two queries were finally abandoned by the TREC committee because they did not get enough time to judge relevant documents for them.), RW+RQLM outperforms QL on 31 queries, performs the same as QL on 2 queries and worse than QL on 15 queries.

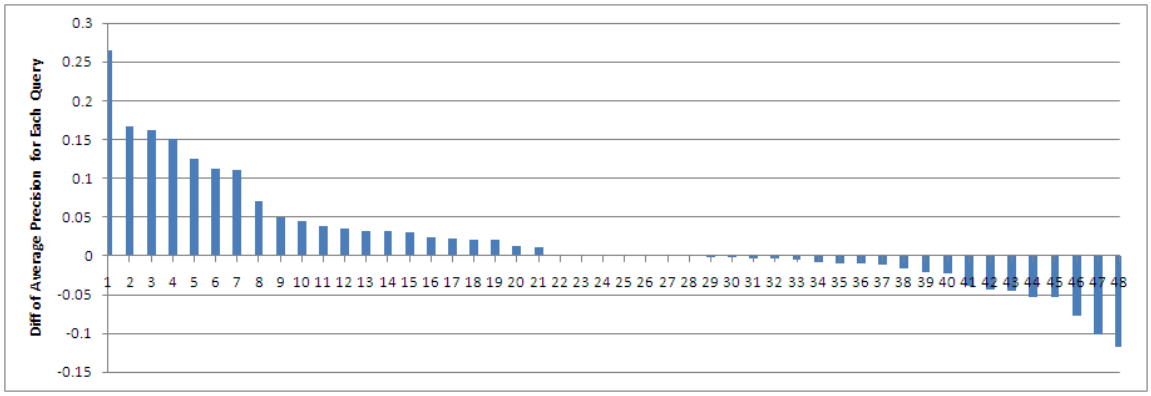


Figure 4.6. The difference of the average precisions (AP) between RW+RQLM and SRM on each individual test query from TREC 2010 Web Track. Above the x-axis reflect queries where RW+RQLM out-performs SRM. Y-axis denotes the difference, computed using (RW+RQLM's AP minus SRM's AP) of each test query. All the differences are sorted then depicted to show the IR performance difference of two approaches. Among 48 queries, RW+RQLM outperforms SRM on 26 queries, performs the same as SRM on 2 queries and worse than SRM on 20 queries.

on each individual test query from TREC 2005 Terabyte Track, respectively. Figure 4.5 and 4.6 show the difference of the average precisions (AP) between RW+RQLM and QL, as well as between RW+RQLM and SRM, on each individual test query from TREC 2010 Web Track, respectively.

We can see from these tables and figures that using the semantic click-through features discovered by different approaches can help to improve web search performance, although performance is affected in different degree by the choice of their model parameters across different query sets. We have the following main observations:

1. Using click-through features extracted from the MS-QLOG benefits web search tasks on the ClueWeb09 data more than the ones on the GOV2 data. This is not surprising because the TREC retrieval tasks on the ClueWeb09 data are, in nature, more similar to real-world web search tasks as those recorded in MS-QLOG: (1) the ClueWeb09 dataset were crawled from the general web while the GOV2 collection was crawled only from government web sites; (2) the queries in the TREC Web Tracks were created to closely simulate the real world web search scenarios, while the queries in the TREC Terabyte Tracks were created to target government web pages in order to have some relevant pages in the GOV2 data, so may be different from the recorded web queries in MS-QLOG.

2. On the test query sets in two retrieval tasks and the training query set in the ClueWeb09 retrieval task, both RQLM and RW+RQLM performed statistically significantly better than QL in terms of MAP and P@10. In the first retrieval task on the GOV2 collection and the second retrieval task on the ClueWeb data, RW+RQLM outperformed QL on 34 queries (68% of 50 test queries) and 31 queries (62% of 48 test queries), respectively. This demonstrates that using click-through query language model features discovered by our content based approach for web pages with no clicks can improve the web search performance significantly. This also indicates that our content based approach can some-

what alleviate the click-through sparseness problem. In addition, RW+RQLM performed slightly better than RQLM on the training query sets in both retrieval tasks and the test query set in the ClueWeb09 retrieval task, indicating that the combination of our content based approach and the click-graph based random walk approach can further reduce click-through sparseness and refine the discovered click-through features for improving search.

3. The query-side information discovery approach (SRM) achieved the best performance on the training query sets in both retrieval tasks. In addition, SRM outperformed RW+RQLM on 28 test queries (56%) in the first retrieval task on the GOV2 collection, although it only outperformed RW+RQLM on 20 test queries (43.5% of the 46 queries where the performance of SRM and RW+RQLM differed) in the second retrieval task on the ClueWeb data. This shows that when the model parameters are carefully tuned, the SRM approach can discover implicit query language information to improve the search effectiveness. $\beta_p = 0.99, \beta_q = 0.01$ in the first retrieval task implies that the extended query field content mainly comes from the content of the MS-QLOG-Web pages that have the highest likelihoods of generating the original query. This situation is similar to the typical relevance model approach (Lavrenko and Croft 2001) except that the training collection and the searched collection are different. In contrast, $\beta_p = 0.01, \beta_q = 0.99$ in the second retrieval task implies that the extended query field content mainly determined by each query's similar queries in the query log and their corresponding clicked pages' content, and that the query log information is more helpful for the second retrieval task on the ClueWeb09 data. However, we observe that on the test queries SRM achieved very little improvement over the QL baseline with the tuned model parameters on the training queries. We will do more analysis on this issue in §4.4.3.3 to investigate some possible causes, such as how sensitive the performance of SRM for

the web search tasks is to the change of its model parameters on different query sets.

To make sure that our demonstration of the effectiveness of using discovered missing semantic click-through features for general web search is not compromised by a weak QL baseline, we show in Table 4.8 some previous results on the TREC 2009 Web Track *ad hoc* search task from some participants (Koolen and Kamps 2010). The 2nd-4th rows of the table show Koolen and Kamps's results on the same retrieval task when they examined the potential of using existing anchor text in large scale web corpora for helping search. One major difference between their QL baseline and ours is that they used linear smoothing approach while we use Dirichlet smoothing which usually performs better than linear smoothing. Comparing Table 4.8 with our results in Table 4.6, we can observe that our baseline performs better than all of Koolen and Kamps's three methods in terms of statMAP. The 5th-7th rows of Table 4.8 show the top3 best official TREC submissions for the same retrieval task from other participants. Comparing these top-performing TREC submissions with our results, we can see that our three retrieval approaches that use click-through query language information discovered from the web query log achieve similar performance to them.

To summarize, our content based approach can effectively discover missing click-through features for web pages with no clicks to help improving retrieval performance. Combining our approach with the random walk approach can further improve the quality of the discovered features from click-through data that have sparseness problem thus further help search. The query-side implicit information discovery approach performs very well on some query sets but not so well on other query sets, depending on whether effective SRMs can be built from the training web pages to better represent information need underlying the queries.

4.4.3.3 More Analysis

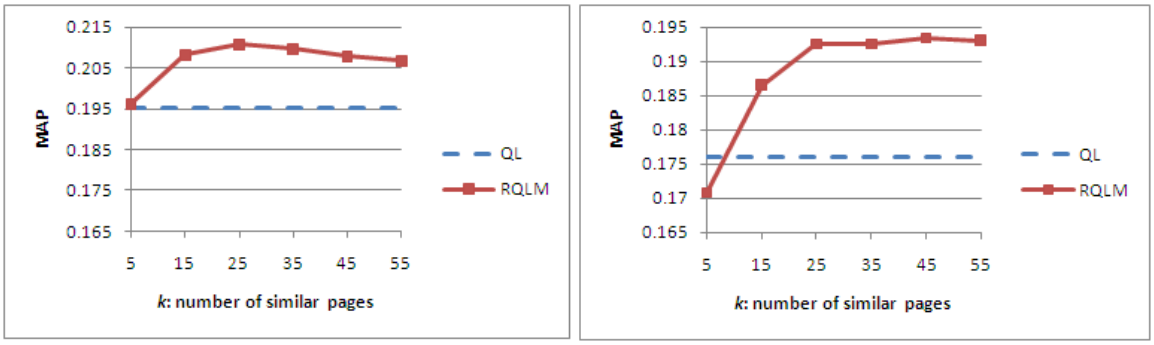
We are concerned about how sensitive different approaches' performance is to the change of their retrieval model parameters. Specifically, for our content based approach (RQLM) and the combination approach (RW+RQLM), we are interested in how many similar pages of each page are needed in order to build RQLMs that can improve retrieval performance the most and how changing this number will affect the retrieval performance. For the combination approach, we are further interested in the impact of using the augmented queries discovered by the random walk approach for helping search. For the SRM based approach, we are concerned with the impact of different number of feedback pages used to build the SRM and the mixture weight between the original query language and the built SRM on this approach's retrieval performance.

As discussed in the previous section, the *ad hoc* web search tasks on the ClueWeb09-T09B collection in our second retrieval task better simulate the real-world web search scenarios; therefore, we use this task to investigate the impact of different model parameters of different approaches on their search performance.

Figure 4.7(a) and (b) depict the model parameter selection's impact for RQLM on the training/testing queries in this retrieval task, respectively, where we fix $\beta = 0.9$ while varying k. Figure 4.8(a) and (b) depict the model parameter selection's impact for RW+RQLM on the training/testing queries, respectively, where we fix $\alpha = 0.01, \beta = 0.9$ while varying γ and k. Figure 4.9(a) and (b) depict the model parameter selection's impact for SRM on the training/testing queries, respectively, where we fix $\beta_p = 0.01, \beta_q = 0.99, N = 100$ while varying λ and k.

From Figure 4.7 and 4.8, we have the following major observations on the two web-side implicit information discovery approaches:

1. Using click-associated queries from about $25 \sim 35$ most similar pages to build RQLM for each page can achieve near optimal retrieval performance on both



(a) training (b) testing

Figure 4.7. The impact of choosing different number (k) of most similar pages on RQLM's retrieval effectiveness. (a) The impact on performance with our training queries; (b) the impact on performance with our test queries.

training/test query sets. Increasing k beyond 35 brings little additional benefit to (or even hurt) the retrieval performance, and only changes the performance very slowly. This property means that in real-world use, for efficiency we need only index click-through information from a small number of similar pages of each page for both approaches, without sacrificing their retrieval effectiveness.

2. Using augmented queries discovered by the random walk approach from the click graph can slightly help the retrieval effectiveness. The mixture weight γ's value can be selected between $0.4 \sim 0.6$ across different query sets and the change of this value among this range has little impact on the retrieval performance of RW+RQLM. This also indicates that click-through features from the augmented queries discovered by the random walk approach are at least as useful as the click-through features from the original click-associated queries for search.

Figure 4.9 shows that the retrieval performance of the SRM based approach mainly depends on whether the training web page collection (which is MS-QLOG-Web here) contains web pages whose content can be useful for discovering implicit field information in the query for searching the target web collection (which is ClueWeb09-T09B

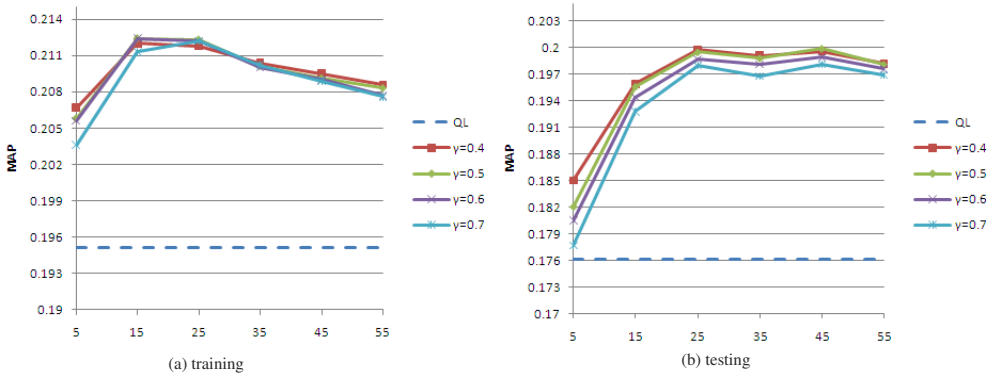


Figure 4.8. The impact of choosing different number (k) of most similar pages and mixture weight γ (between the original click-associated query language model and the language model from the augmented queries discovered by the random walk approach) on RW+RQLM 's retrieval effectiveness. (a) The impact on performance with our training queries; (b) the impact on performance with our test queries.

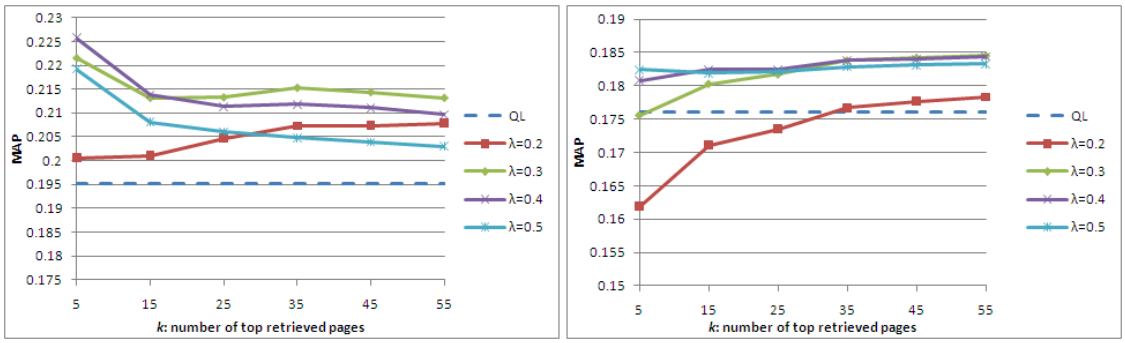


Figure 4.9. The impact of choosing different number (k) of retrieved pages to build SRM and mixture weight λ (between the built SRM and the original query language model) on SRM's retrieval effectiveness. (a) The impact on performance with our training queries; (b) the impact on performance with our test queries.

here). For the training queries, using only top-5 feedback pages for extending the *Page Content* field of each query can achieve very good performance (even better than the RW+RQLM approach); in contrast, for the test queries, the performance improvement over the QL baseline is very little and achieved by using more feedback pages for building SRM. Furthermore, the choice of the mixture weight λ also affects SRM's retrieval performance significantly, both within the same query set and across different query sets. The choice of the best λ indicates the quality of the built SRM for different query sets: the higher quality is the SRM, the smaller λ and less information from the original query language model are needed to reconstruct query fields for search. To summarize, compared with our content based approach and the combination approach, the SRM based approach's retrieval performance is more sensitive to the selection of the model parameters across different query sets for this retrieval task.

4.5 Conclusions

In this chapter, we employed our general perspective of discovering implicit information for search in Figure 1.1 (in Chapter 1) to address the click-through data sparseness issue when using web query logs for search. We presented and compared webpage-side and query-side approaches of discovering plausible semantic click-through features from web query logs for web search.

For the webpage-side approach (in Figure 4.2(a)), we proposed a language modeling based method that uses web content similarity for discovering plausible click-through features for web pages with no or few clicks. Similar to our approach of discovering anchor text in Chapter 3, here we computed a relevant (click-associated) query language model, called RQLM, from the click-associated queries of the similar pages of a web page in the web query log for discovering the page's plausible (but missing) click-through features. Compared with the random walk approach (Craswell

and Szummer 2007), the RQLM approach does not need to use specific click graph structure to discover semantically related queries for pages thus can handle the pages with no clicks and further mitigate click-through sparseness. Compared with the Good-Turing based smoothing approach (Gao *et al.* 2009), our approach can discover different semantic click-through features for web pages having different content and no clicks in the query log. Moreover, we presented a combination approach that takes advantage of both the random walk approach and our content based approach to further reduce click-through sparseness and improve the quality of discovered click-through features for search. We then described how we use discovered information for web search by using the mixture model (Nallapati *et al.* 2003; Ogilvie and Callan 2003) in the language modeling retrieval framework.

For the query-side approach (in Figure 4.2(b)), we presented how we adapted the Structured Relevance Model (SRM) for this task where we used the click-through information in the web query log to discover plausible semi-structured query field information for search. The basic procedure is similar to our query-side approach of handling anchor text information in §3.4.2 – adding some structure to an unstructured query, viewing both queries and web pages as semi-structured records and using the SRM approach for search – except that the training web pages and the searched target here are different web collections.

We evaluated the retrieval performance of the above two approaches (webpage-side and query-side) with two different sets of *ad hoc* web search tasks. The first one consisted of the retrieval tasks in the TREC 2004-2005 Terabyte Tracks performed on the GOV2 collection and the second one consisted of the retrieval tasks in the TREC 2009-2010 Web Tracks performed on the ClueWeb09-T09B collection. The results on both sets of web search tasks showed that discovering click-through features for web pages with no clicks can help to improve the web search performance statistically significantly, compared with the retrieval baseline that does not use this information.

The webpage-side discovery approaches, including RQLM and the combination approach (RW+RQLM), for this IR challenge performed robustly across different query sets while the query-side discovery approach's retrieval performance was more sensitive to the selection of the model parameters on different query sets. In addition, the click-through features discovered by the random walk approach complemented those discovered by our content approach for helping search and the combination approach performed slightly better than the content-only approach on three of the four query sets in our experiments.

There are several interesting directions of future work. It seems worthwhile to explore using the discovered semantic click-through features beyond the language modeling based retrieval framework. For example, we can use those features in the learning-to-rank retrieval approach (Burges *et al.* 2005), so that different approaches described here may be combined with the Good-Turing based smoothing approach (Gao *et al.* 2009) to achieve better retrieval performance. Moreover, here we only explored using the contextual translation probability $p(P_i|P_0)$ between web pages to discover useful missing *semantic* click-through features. However, theoretically, we can also use this probability to compute an expected feature $E(f_{P_0}) = \sum_{f_{P_i} \in \mathcal{F}} f_{P_i} \times p(P_i|P_0)$ for any feature of a page P_0, using the same click-through feature f_{P_i} of P_0's similar pages P_i. In this way, we can compute an expected feature value that can incorporate web content similarity information to help search. Similar to Gao et al.'s approach, this approach also aims to smooth the click-through features for web pages with no clicks, but leverages the web content similarity during the smoothing. We would like to explore the utility of these smoothed click-through features for retrieval.

CHAPTER 5

DISCOVERING IMPLICIT GEOGRAPHIC INFORMATION IN WEB QUERIES

5.1 Introduction

In this chapter we address an implicit information discovery challenge where a user searches for information associated with a particular geographic location (city in our case) but omits the location name when formulating the query. For example, when the user issues the query "eiffel tower tour", he or she probably wants travel information around *Paris, France*. Our goal is to detect such queries and predict the plausible city missing in them (e.g. Paris, France in the above query example). Again, we address the information discovery issue here using the general perspective of discovering implicit information for IR in Figure 1.1. We start with a detailed description of the background of this research issue.

Previous research has shown that more than 13% of web queries contain explicit geographic (referred to as "geo" for simplicity) information (Jones *et al.* 2008; Sanderson and Kohler 2004; Welch and Cho 2008). Identifying geo information in user queries can be used for different retrieval tasks: we can personalize retrieval results based on the geo information in the query and improve a user's search experience; we can also provide better advertisement matching and deliver more information about the goods and services in some specific geographic areas to the potentially interested users. Previous research has demonstrated how to improve retrieval performance for a query by incorporating related geo information when this information explicitly appears in the query or is known beforehand (Andrade and Silva 2006; Yu and Cai 2007; Jones *et al.* 2008).

However, recent research has found that only about 50% of queries with *geo intent* – i.e., queries where the users expected the results to be contained within some geographic radius – had *explicit* geo location names (Welch and Cho 2008). For example, many users input the query "space needle", expecting the search engine to automatically detect their intent to find relevant travel information in Seattle. Therefore, identifying implicit geo intent and accurately determining location information is important and necessary for any retrieval model that leverages geo information. We expect that in handheld devices like cell-phones, the percentage of queries with implicit geo intent will be much higher. For convenience, we refer to *geo intent queries* as *geo queries* in the rest of this chapter.

In our research, we consider detecting implicit geo queries and discovering their plausible geo information at a fine grained city level. Previous research has shown that a large portion (84%) of explicit geo queries contain city level information (Jones *et al.* 2008), which implies that users often have a city level granularity in mind when issuing geo queries. We therefore believe that finding implicit city level information can greatly help satisfy users' specific geo information needs, e.g. a user who searches for "macy's parade hotel rooms" can receive a variety of information about hotels in New York City. For the convenience of description, we consider that an explicit geo query consists of (a) *a location part*: that explicitly helps identify the location and (b) *a non-location part*. For example, in the query "pizza in 95054", the term "95054" is the location part and the remaining terms, the non-location part[1].

We hypothesize that *implicit* geo queries may be similar in content to the non-location part of *explicit* geo queries and that the city level information in the *implicit* geo queries corresponds to the location part of their similar *explicit* geo queries.

[1]The word "in" will be removed from the non-location part of geo queries as a stopword after the data preprocessing steps described later in §5.3.

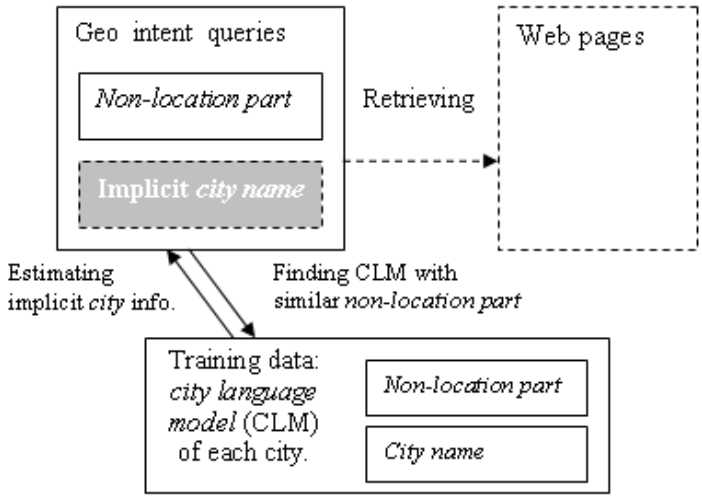


Figure 5.1. The specific perspective of discovering implicit city information in location-specific web queries

Under this assumption, we develop language modeling based techniques for implicit geo detection and missing city level geo information prediction.

Figure 5.1 illustrates our approach. Specifically, we build query language models for different cities (called *city language models* or CLMs) from the *non-location* part of the training explicit geo queries. Then we calculate the posterior of each city language model generating the observed query string (non-location part of the query), and then utilize the posteriors to detect implicit geo search intent and predict plausible city information for a query. As we mentioned in §1.5, because previous research has explored how to incorporate explicit geo information of queries into retrieval models (Andrade and Silva 2006; Yu and Cai 2007; Jones *et al.* 2008), here we only consider finding implicit geo queries and their plausible missing city-level geo information. Accordingly, we show the retrieval part in Figure 5.1 with the dashed line.

In order to be able to accurately train language models for thousands of different cities, we utilize a large sample from a months' worth of web search logs from a major search engine (Yahoo!) which contains more than 2.8 billion search instances. We

also use this Yahoo! query log sample to design experiments and generate *simulated* implicit geo queries to evaluate the performance of our approach.

The remaining parts of this chapter will be organized as follows. We begin by reviewing related work in §5.2. Next, in §5.3, we describe how we build city language models for implicit geo search intent detection and missing city information prediction. Then we describe the experimental setup, present and discuss the evaluation results in §5.4. We conclude in §5.5.

5.2 Related Work

Although considerable work has been done on how to utilize geographic information in meta data for IR (Gey *et al.* 2005; Gey *et al.* 2006; Mandl *et al.* 2007; Mandl *et al.* 2008; Purves and Jones 2007), research on automatically detecting and understanding users' geo search intent in web search has just started. In 2007, the GeoCLEF (Cross-Language Geographical Information Retrieval in Cross-Language Evaluation Forum) workshop began a geo query parsing and classification track (Mandl *et al.* 2007), which required participants to not only extract location and non-location information of explicit geo queries but also required them to classify the non-location part into three predefined sub-categories: informational (e.g. news, blogs), yellow pages (e.g. restaurants, hospitals) and maps (e.g. rivers, mountains). Different from the research focus in this workshop, our work aims at detecting users' *implicit* geo intent in the queries. Welch and Cho's pilot study (2008) shows that features extracted from non-location parts of explicit geo queries can help discriminate queries that have geo intent from those that don't. Different from their work, we utilize more complex language modeling features for not only detecting users' implicit geo intent but also discovering the plausible (but missing) location information from the query content.

Raghavan et al. (2004) built language models from the contextual language around different name entities (e.g. person, location, organization, etc) in a TREC corpus, and utilized these entity language models for linking, clustering and classifying different entities. Pasca (2007) utilized different contextual language patterns in the search logs to extract different types of name entities. These works demonstrated the effectiveness of using contextual features for categorizing entities. In our work, we build language models for geo location entities from large scale web search logs, and investigate whether more complex contextual features can help discover users' implicit geo search intent and the related missing locations.

Besides using web search logs, some research considers mining both top web search results and web search logs to disambiguate whether a query that contains a geo location name implies geo intent or not – e.g. determining whether the query "New York Style cheesecake" is a geo query (Wang *et al.* 2005) . This work complements our approach to better understand users' implicit specific geo intent.

Our research on discovering and analyzing implicit geo search intent has also been described in one published paper (Yi *et al.* 2009).

5.3 Detecting Implicit Geo Intent and Predicting Implicit City Information

We formally describe how we build city-level query language models (city language models or CLMs) from web query logs for the implicit geo search analysis in this section. For the convenience of discussing our approach, we start by briefly describing how we obtain the training data (queries that contain explicit city information), which correspond to the information in the bottom box shown in Figure 5.1 (in the introduction of this chapter).

First, for each query Q in the web search log, we correct possible spelling errors and remove any stopwords present in the INQUERY (Broglio *et al.* 1993) stopword

list[2]. We do not apply stemming on the queries because previous research showed that stemming has little impact on implicit geo intent detection while removing stopwords has significant positive impact (Welch and Cho 2008).

Then we utilize a geo location analysis tool (JONES *et al.* 2008; RIISE *et al.* 2003) to identify every possible *explicit* city-level geo query Q_{cg}. This tool utilizes both context-dependent (e.g. 'in','at') and context-independent features to find possible location parts in a query, and maps these location parts to a large global location databases containing zip-codes, cities, counties, states, countries etc. This tool calculates a confidence score in the range (0,1) for each location candidate identified in the query based on the confidence of whether the candidate is indeed a geo location, and outputs all the possible locations and confidence scores. In our research we only consider location candidates whose confidence scores exceed 0.5.[3]

In the above way, we obtain explicit city level geo queries Q_{cg} from query logs and decompose each of them as (Q_c, Q_{nc}), where Q_c and Q_{nc} denote the location/city and non-location part respectively. Next, we build a CLM for each city by using all the identified non-location portions (Q_{nc}) of the queries that have the same city name in their location part Q_c.

5.3.1 City Language Models

City names often have strong co-occurrence statistics with terms or phrases like 'map', 'hotel', 'hospital' and so on in the query logs. Therefore, analyzing the language used in the non-city parts (Q_{nc}) that co-occur with a certain city name in the location part (Q_c) can possibly help discover missing city information in an implicit geo query.

---

[2]We remove 'ff', 'first', 'stave' and 'staves' from the original version and use the remaining 414 stopwords.

[3]To limit our scope we only consider U.S. city locations.

To build language models for each city, we go beyond the "bag of words" approach used in entity language models built by Raghavan et al. (2004) and instead follow a bigram language model approach. The reason is that bigram information can be very important to infer implicit geo intent from phrases. For example, the words 'times' and 'square' individually may not imply geo intent, but the phrase 'times square' has a high possibility of being related to New York City. We do not build trigram language models because trigrams in web queries are much sparser than bigrams, making trigram language models not as robust as bigram language models. In the typical bigram language modeling approach, the probability of a string is expressed as the product of the probabilities of the words that compose the string, where the probability of each word is conditioned on the identity of the previous word (Chen and Goodman 1996); therefore, given a query $Q = w_1 \cdots w_n$, we have:

$$P(Q) = P(w_1) \prod_{i=2}^{n} P(w_i|w_1^{i-1}) \approx P(w_1) \prod_{i=2}^{n} P(w_i|w_{i-1}), \tag{5.1}$$

where w_i^j denotes the string $w_i \cdots w_j$. Then, for each city C_k, we build bigram language models from the non-location portions (Q_{nc}) of all the explicit geo queries (Q_{cg}) that have the location portion (Q_c) identified as the city C_k. In this way, we can calculate the probability $P(Q|C_k)$ of a query Q generated from a city C_k's language model by:

$$P(Q|C_k) = P(w_1|C_k) \prod_{i=2}^{n} P(w_i|w_1^{i-1}, C_k) \approx P(w_1|C_k) \prod_{i=2}^{n} P(w_i|w_{i-1}, C_k). \tag{5.2}$$

Researchers have proposed a broad range of smoothing techniques that adjust the maximum likelihood estimation (MLE) of parameters to solve the zero-frequency problem in language modeling, and thereby produce more accurate estimations and predictions. Many thorough comparison studies of different smoothing techniques can be found in the literature (Chen and Goodman 1996; Zhai and Lafferty 2001b). Different smoothing techniques can have significantly different results. In this study, for the estimation of bigram probability, we employ a state-of-the-art smooth-

q ="Disney world ticket"		q ="Harvard University"	
City Name	$P(C_i\|Q)$	City Name	$P(C_i\|Q)$
Orlando	0.98011	Cambridge	0.63545
Kissimmee	0.01386	Princeton	0.05360
Anaheim	0.00240	Longwood	0.05334
New Castle	0.00135	Boston	0.01979
San Antonio	0.00044	Tuskegee	0.01719

Table 5.1. Top-5 cities and the city generation posteriors for two sample queries.

ing technique (method B in Chen and Goodman's study(1996)), which combines two intuitions from Dirichlet smoothing and Good-Turing smoothing:

$$P(w_i|w_{i-1}, C_k) = \frac{\#(w_{i-1}^i, C_k) + \alpha P(w_i|C_k)}{\#(w_{i-1}, C_k) + \alpha}, \alpha = \beta \times |V_{C_k}|, \tag{5.3}$$

where $\#(w_i^j, C_k)$ and $\#(w_i, C_k)$ denotes the frequency counts of the string w_i^j and character w_i (respectively) in the non-city parts (Q_{nc}) related to the city C_k, $|V_{C_k}|$ denotes the vocabulary size of the words that appear in the city C_k's language model, α is the Dirichlet smoothing parameter, β is a constant to control the degree of smoothing for different cities that have different vocabulary sizes. For the unigram probability $P(w_i|C_k)$ in equation 5.3, we employ standard Dirichlet smoothing:

$$P(w_i|C_k) = \frac{\#(w_i, C_k) + \gamma P(w_i|C_\bullet)}{\#(w_\bullet, C_k) + \gamma} = \frac{\#(w_i, C_k) + \gamma \#(w_i, C_\bullet)/\#(w_\bullet, C_\bullet)}{\#(w_\bullet, C_k) + \gamma}, \tag{5.4}$$

where $w_\bullet$ denotes all the words and $C_\bullet$ denotes all the cities, e.g. $\#(w_\bullet, C_k)$ denotes the counts of all the words appearing in the non-location parts of geo-intent queries (Q_{nc}) related to the city C_k and $\#(w_\bullet, C_\bullet)$ denotes the counts of all the words co-occurring with all the cities. γ is the Dirichlet smoothing parameter.

To predict the city information relevant to an implicit geo query Q, we calculate the posterior probability of each query Q generated from a city C_i by:

$$P(C_i|Q) \propto P(C_i)P(Q|C_i), \tag{5.5}$$

where we set the prior $P(C_i)$ to be a uniform distribution, i.e. the posterior calculation will be only affected by the city generation probability $P(Q|C_i)$, and not be biased towards those cities that appear most frequently in the query logs. After calculating all the posteriors, we can sort them to discover the most probable cities that each implicit geo query Q may be generated from. Table 5.1 shows the top-5 cities and the corresponding posteriors calculated by our CLMs, trained in experiments, for two sample queries: "Disney world ticket" and "Harvard University". 'New Castle' appears in the top cities related to the first query because of its ambiguous meaning – the geo analysis tool we used fails to determine whether it means a new palace in Disney or the city named 'New Castle'.

Later in §5.4, we present how we design experiments to evaluate the above approach to predict plausible city information for a large set of synthetic implicit geo queries.

5.3.2 Detecting Implicit Geo Intent with Geo Language Features

In order to use a rich set of geo information related language features for detecting implicit geo queries, we consider a *discriminative* machine learning approach, i.e. we train a discriminative classifier that uses geo language features to classify each query into two categories: with geo search intent and without.

The posteriors ($P(C_i|Q)$) of city language models generating queries, described in the previous section (§5.3.1), are useful as features to detect implicit city level geo intent. We use them as part of the geo language features for building the discriminative classifier. In addition, we extract a set of geo information features directly related to the n-grams that occur in the non-city parts (Q_{nc}) of explicit geo queries (Q_{cg}) for building the classifier. Intuitively, the n-grams in the Q_{nc} of Q_{cg} can help detect users' implicit geo intent, e.g. the queries "golden gate bridge" or "fishermen's wharf" may imply that users are interested in information about San Francisco. Thus, we

view each unigram, bigram and trigram in the non-location portions (Q_{nc}) of all the explicit geo queries (Q_{cg}) as a *Geo Information Unit* (GIU) that can help discover users' implicit geo intent, and extract statistics in the training data for each information unit. Then given any new input query Q, we find all the GIUs in this query and utilize them to generate a wide range of features for implicit geo intent detection.

For each n-gram GIU $w_i^{i+n-1} = w_i \cdots w_{i+n-1}$ appearing in the non-location part (Q_{nc}) of all geo-intent queries (Q_{cg}), we calculate the following GIU features:

- The frequency count of w_i^{i+n-1} in the set of queries, Q_{nc}, from all cities $C_\bullet$, denoted as $\#(w_i^{i+n-1}, C_\bullet)$, and the MLE probability ($P_g(w_i^{i+n-1})$) of w_i^{i+n-1} appearing in the n-grams of all the queries, Q_{nc} : $P_g(w_i^{i+n-1}) = \#(w_i^{i+n-1}, C_\bullet)/\#_g(ngrams)$, where $\#_g(ngrams)$ denotes the number of n-grams in the set of all Q_{nc}.

- The frequency of w_i^{i+n-1} in all queries (including both geo and non-geo intent), denoted as $\#(w_i^{i+n-1})$, and the MLE probability of w_i^{i+n-1} appearing in the n-grams of all the queries: $P(w_i^{i+n-1}) = \#(w_i^{i+n-1})/\#(ngrams)$, where $\#(ngrams)$ denotes the number of n-grams in all the queries.

- The pair-wise mutual information (PMI) score (Van Rijsbergen 1979) between w_i^{i+n-1} and all city locations $C_\bullet$: $PMI(w_i^{i+n-1}, C_\bullet) = \frac{P(w_i^{i+n-1}, C_\bullet)}{P(w_i^{i+n-1})P(C_\bullet)} = \frac{P_g(w_i^{i+n-1})}{P(w_i^{i+n-1})}$.

- The number of cities that co-occur with w_i^{i+n-1}.

- The MLE probability $P(w_i^{i+n-1}|C_k)$ of w_i^{i+n-1} appearing in the n-grams of Q_{nc} that co-occur with city C_k, calculated by: $P(w_i^{i+n-1}|C_k) = \frac{\#(w_i^{i+n-1}, C_k)}{\#_{C_k}(ngrams)}$, where $\#_{C_k}(ngrams)$ denotes the number of n-grams in the Q_{nc} that co-occur with city C_k.

- Given the MLE probability $P(w_i^{i+n-1}|C_k)$ we calculate the posterior: $P(C_k|w_i^{i+n-1}) \propto P(C_k)P(w_i^{i+n-1}|C_k)$, where we assume $P(C_k)$ is a uniform distribution. Then we

find the city C_m that has the maximum posterior to generate w_i^{i+n-1}, and use $P(C_m|w_i^{i+n-1})$ and the frequency counts $\#(w_i^{i+n-1}, C_m)$ as two more GIU features.

- To measure the skewness of the posteriors $\{P(C_k|w_i^{i+n-1}),\ k = 1, \cdots, N(w_i^{i+n-1})\}$, where $N(w_i^{i+n-1})$ denotes the number of cities that co-occur with the GIU, w_i^{i+n-1}, we calculate the K-L divergence between the posteriors and a uniform distribution $U(w_i^{i+n-1}) = 1/N(w_i^{i+n-1})$ and is computed by the following formula: $\sum_{k=1}^{N(w_i^{i+n-1})} P(C_k|w_i^{i+n-1}) \log \frac{P(C_k|w_i^{i+n-1})}{1/N(w_i^{i+n-1})}$.

After calculating the above features for each GIU, given a new query Q, we extract all the GIUs in it and then utilize the features of these GIUs to form a high dimensional sparse feature vector for representing this query. These feature vectors are then used as input for training the discriminative classifier to detect users' implicit geo search intent.

5.4 Evaluation Experiments

We consider two implicit geo search intent analysis tasks: (Task I) detecting whether a query containing no explicit geo information has geo intent and (Task II) discovering plausible missing city information in implicit geo queries. We design two set of experiments to evaluate the performance of our proposed CLMs for Task I and the posteriors from the CLMs and the additional geo-related language features (GIU features) for Task II. Next in §5.4.1, we briefly describe the query log data we used to generate the evaluation data in the experiments. Then we describe, in §5.4.2 and §5.4.3, how we generated labeled data, the classifiers we used and some preliminary results, in Task I and II, respectively.

5.4.1 Overview of Data

We utilize a large industrial-scale real-world web search log from Yahoo! for this study. The *training set* is a subset of the Yahoo! web search log during May, 2008. It

City Name	Frequency in geo sub training set	Frequency in geo sub testing set
New York	3794960	3865216
Los Angeles	3207062	3228888
Chicago	2275231	2397036
Houston	1929131	1926341
Las Vegas	1755695	1794026

Table 5.2. Statistics of top-5 most frequent cities in two geo query subsets.

contains about 2.13 billion rows of search instance records covering about 1.44 billion queries and related information, e.g. users' IP and the clicked URLs. The *testing set* is randomly sampled from the Yahoo! web search log during June, 2008 and contains about 2.10 billion rows of search instance records covering about 1.42 billion queries and related information. We applied an explicit geo information analysis tool (Jones *et al.* 2008; Riise *et al.* 2003) on both the training and the testing sets to identify each explicit geo query that contains a U.S. city location. In this way, about 96.2M U.S. city level geo queries are identified in the training set and extracted to form a *geo training subset*, and about 96.7M U.S. city level geo queries are identified in the testing set and extracted to form a *geo testing subset*. We find 1614 distinct cities in the two geo query subsets. Table 5.2 shows the 5 most frequent cities in the geo training/testing subset, respectively.

We build bi-gram language models for each city by using the non-location parts Q_{nc} of all the explicit geo queries $Q_{cg} = (Q_c, Q_{nc})$ in the geo training subset. Then given any implicit geo query Q, we can calculate a set of city generation posteriors $P(C_i|Q)$ from the extracted language models of each city C_i. We then use the posteriors as part of the geo related language features for building the classifier to detect implicit geo intent. In experiments we use the 10 largest posteriors of each query as features for simplicity and noise reduction.

We also utilize all of the original training set and the geo training subset to extract additional n-gram GIU features (described in §5.3.2) for all the unigrams, bigrams

and trigrams that appear in the non-location parts (Q_{nc}) in the geo training subset. These features are also used for building the geo intent detection classifier.

5.4.2 Discovering Implicit Geo Intent in Queries

We use the URLs that have been frequently clicked for a query to automatically generate geo/non-geo intent labels for queries, for saving human-labeling efforts. For example, if many users repeatedly clicked the URL `local.yahoo.com` for a query, it has a high probability of having geo intent.

To find URLs that reliably imply users' geo intents, we consider only the domain name (DN) of the URL. We collect 100 DNs that are most frequently clicked for queries in the geo training subset to form the set DN_1. We also collect 100 DNs that are most frequently clicked from the other queries that are not in the geo training subset but in the whole training set, into another set DN_2. Then we obtain the DN sets DN_+ and DN_- for labeling queries that may/may not have geo intent by:

$$DN_+ = DN_1 \backslash DN_2, \quad DN_- = DN_2 \backslash DN_1.$$

Some DNs that are intuitively useful for labeling users' geo intent and appear in both DN_1 and DN_2 end up being excluded from both DN_+ and DN_-. On analysis we found a few possible reasons for this. For example, in the above process, the clicked URLs of possible implicit geo queries or larger regional level (state/country) geo queries are counted in DN_2. Similarly, the clicked URLs of some ambiguous queries where the explicit geo location analysis tool (Riise *et al.* 2003) falsely identifies city names are counted in DN_1. Therefore we introduce weak supervision into this domain name selection process by putting three useful DNs back to DN_+ and two back to DN_-:

$$DN_+ = DN_+ \cup \{\text{www.citysearch.com},$$
$$\text{www.yellowpages.com}, \ \text{local.yahoo.com}\}$$
$$DN_- = DN_- \cup \{\text{en.wikipedia.org}, \ \text{answers.yahoo.com}\}$$

DN_+	DN_-
www.local.com	search-desc.ebay.com
travel.yahoo.com	www.youtube.com
www.tripadvisor.com	www.amazon.com
www.yellowbook.com	www.myspace.com
www.city-data.com	www.nextag.com

Table 5.3. Some DNs in DN_+ or DN_-

In this way, we obtain 67 DNs in DN_+ and 64 DNs in DN_- respectively. Some example DNs from the two sets are shown in Table 5.3.

For any query in the geo training subset, if it has a clicked DN in DN_+, we label the query as a positive sample. For any query that is in the training set but not the geo training subset, if it has a clicked DN in DN_-, we label the query as a negative sample or non-geo intent query. We remove duplicates that have the same query terms and domain names. After that, we obtain 7.5M positive and 57.8M negative samples. We then use the location portion (Q_c) of the positive samples as the labels and *the non-location portion (Q_{nc}) as the implicit geo queries*. Note that this way of generating implicit geo queries is artificial and the generated ones may have different properties from the truly implicit geo queries. For example, the non-location part 'map' of an explicit geo query 'new york city map' is not a normal implicit geo query issued by the web user. Nevertheless, many synthetic implicit geo queries have similar properties as the truly ones (e.g. "space needle" instead of "space needle seattle" may be issued by the web searcher), and this approach enables us to avoid the prohibitively expensive human-labeling procedure of obtaining large amounts of real implicit geo queries from millions of queries in web query logs for evaluation.

Next, we randomly sample 20,000 simulated implicit geo queries and 20,000 non-geo queries to obtain 40,000 queries in the *training subset I*. For evaluation, we generate two testing subsets: *testing subset I-1* and *testing subset I-2* from the original testing set in two ways.

The first method is to follow the same above procedure: labeling positive samples only from queries in the *geo testing subset* that have clicked DNs in DN_+ and extracting Q_{nc} as the implicit geo queries; labeling negative samples only from queries not in the geo testing subset that have clicked DNs in DN_-. In this way, we obtain 8.0M implicit geo queries and 58.1M non-geo queries. Then we randomly sample 80,000 queries (half positive, half negative) as the *testing subset I-1*.

The second method differs from the first in how it finds the positive samples and creates the implicit geo queries. In this method, we directly label both positive and negative samples from the original whole testing set by only checking whether they have clicked DNs in DN_+ or DN_- and without considering whether or not they are in the *geo testing subset*. We use the explicit geo location analysis tool (Riise *et al.* 2003) to find and remove all the possible location portions (place names, zip-codes etc) in the positive and negative samples. Then we remove the duplicates. In this way, we obtain 31.3M positive samples and 53.2M negative samples. Then we randomly sample 80,000 queries (half positive, half negative) as the *testing subset I-2*. Note that classifying testing subset I-2 is more representative of the true query log, and possibly harder, because positive samples are directly obtained from the original testing set instead of only from the *geo testing subset*. Testing subset I-2 may contain some real implicit geo queries instead of only the queries (Q_{nc}) from explicit geo queries as in testing subset I-1.

We use our geo language model features and geo-related language features to train a state-of-the-art classification technique called Support Vector Machines (SVM) (Chang and Lin 2006) for this learning task. We employed a linear kernel (SVM-Linear) as well as a non-linear RBF gaussian kernel (SVM-RBF). Training SVM-linear typically takes much less time than training SVM-RBF, while SVM-RBF usually performs better when the original input feature space is low dimensional.

	Testing subset I-1			Testing subset I-2		
	P	R	Acc	P	R	Acc
SVM-linear	99.9%	66.0%	83.0%	99.9%	48.8%	74.4%
SVM-RBF	98.5%	62.8%	80.9%	97.8%	48.0%	73.5%

Table 5.4. Performances of discovering users' implicit city level geo intent on the testing subset I-1 and I-2 by using SVM. Precision, Recall and Accuracy are denoted by P, R and Acc, respectively.

For each labeled query sample, we calculate the geo language model features – top-10 city generation posteriors and the GIU features – then combine them for classification. We separately scale each feature dimension to be in the range [0,1] for all the samples, and train the classifier based with the data in the training subset I. We employ 5-fold cross validation to select the model parameters that achieve the highest average accuracy. Then we test the optimized classifier on both the testing subset I-1 and I-2.

Performance is evaluated by using the typical precision, recall and accuracy metrics: precision measures the percentage of true positive samples (true geo queries) in the queries labeled by the classifier to be positive (have geo intent); recall measures the fraction of the true positive samples detected by the classifier in all the true positive samples; accuracy measures the percentage of the correct labels, including both positive and negative ones, in the test set. In this task, low precision will hurt users' search experience more than low recall or low accuracy. Thus a classifier for this task in a practical system should have high precision and reasonably good accuracy and recall.

The evaluation results are shown in Table 5.4. Results on both testing sets show that using our proposed geo language features, including the city language model based features and GIU features, to train discriminative classifier can effectively detect implicit geo search intents in the queries with high precision and reasonably good accuracy and recall. For the harder classification task on the testing subset I-2 which better simulates real-life implicit geo queries, our approach can still achieve very high

precision, which is important for users' satisfaction, although the recall and accuracy rates drop noticeably.

As we know, the same web query can be issued by different users at different time. Thus the web log samples from two different months may have considerable amount of the identical queries. We do an overlap analysis in order to better understand our evaluation results. We find that in the 96.7M geo testing subset (from the June sample), about 67% of the queries have appeared in the geo training subset (from the May sample). There are 28.9M and 29.2M distinct queries (Q_{nc}) in the geo training and testing subset, respectively. We find about 48.06% of these distinct queries (Q_{nc}) of the geo testing subset have appeared in the geo training subset. The overlap also reveals that many geo language patterns found in old web query logs can be reused because many geo queries appear repeatedly. This process of splitting the training and test sets by time is a common procedure in domains where the data occurs as a time series [4]. In addition, there are plenty of new geo-queries, revealing that our models can generalize well for new queries as well.

5.4.3 Predicting Implicit City Information in Geo Queries

In this task we aim to predict plausible city information for the implicit geo queries which may contain certain entity that is in some way specific to some particular city. Such "localized entities" may be hotels, local TV and radio channels, local newspapers, universities, schools, people names like doctors, sports teams and so on. Basically if a location (city level) can be pinpointed to some item mentioned in the query, then we say this query is *location-specific*. Examples of a location specific query and corresponding locations are shown in Table 5.5.

[4]http://projects.ldc.upenn.edu/TDT/

Location-specific query	location
airport check metro airport	Detroit
woodfield mall jobs	schaumburg
utah herald journal classified ads	Logan
wkrn news 2	Nashville
motel near knotts berry farm california	Buena Park

Table 5.5. Example of correct predictions of the city name for a location specific query

5.4.3.1 Label Generation

We evaluate our CLMs for retrieving cities in location-specific queries in this experiment. One important property of location-specific queries is that although explicit geo information is missing, one may still accurately discover the exact location (city level) in the user's mind. For example, "Liberty Statue" or "Disney fl" can be viewed as location-specific queries, which are highly likely to be related to New York or Orlando respectively. Our low-cost training method utilizes the non-city part (Q_{nc}) of explicit geo queries as *simulated* implicit geo queries, and tries to discover plausible location-specific queries from them. This approach has another advantage that the city part (Q_c) can be used as the ground truth city label for automatic evaluation. It is extremely expensive to hire human editors to examine over hundred million implicit geo-queries (Q_{nc}) with their city labels (Q_c) and identify all the possible location-specific queries to create training and testing data. Therefore, we utilize the following weakly supervised approach combined with the CLMs for this discovery task, and then sample outputs of the CLMs on the testing data for human evaluation.

Our weakly supervised approach involves designing a few *ad hoc* rules to find the GIUs that may come from location-specific queries. For example, we require that the maximum city generation posterior $-P(C_m|w_i^{i+n-1})$ – be larger than a threshold, t_1, and the corresponding maximum frequency count, $\#(w_i^{i+n-1}, C_m)$, be larger than a threshold t_2; as another example of our rules, we either require that w_i^{i+n-1} appear in less than a threshold, t_3, number of cities or its overall counts in the geo queries

divided by the number of city: $\#(w_i^{i+n-1}, C_\bullet)/\#(|C_\bullet|)$ is larger than a threshold t_4. These rules are constructed by considering the characteristics of the GIU features that location-specific queries may have, and the thresholds are set by looking through the GIUs (w_i^{i+n-1}) and their GIU feature values in the training data. We leave the question of how to automatically generate these rules for future work. In this way, from the *geo training subset* we obtain 1022 unigram GIUs, 4374 bigram GIUs and 3765 trigram GIUs that may come from location-specific queries. We then select queries which contain any of these GIUs in the *geo training/testing subsets*. In this way we form *training subset II/testing subset II*, each of which contains about 1.06M and 1.05M simulated distinct *possible* location-specific queries (distinct Q_{nc}) respectively. We use these automatically generated training and testing subsets to tune parameters for our task. We now describe how to utilize CLMs to further discover cities for location-specific queries from these two subsets.

5.4.3.2 City Language Models for Retrieving Candidate locations

Discovering likely related cities for location-specific queries can be viewed as a challenging multi-category classification task, in which there are 1614 different categories (city labels). Given a query (Q) which has implicit geo-intent and is location-specific, we calculate the city generation posterior $P(C_k|Q)$ of each city C_k by using the CLM and equation 5.5. Then we sort these posteriors and get the corresponding ranked list of cities. We check whether the maximum posterior $P(C_m|Q)$ is larger than a threshold t_a: if yes, C_m is suggested as a candidate location for the location specific query Q. Next, we discuss how to tune t_a with the training subset II.

We utilize the city part (Q_c) as the ground truth city label for each query (remember that the implicit geo-query, Q, is the non-city part, Q_{nc}, of a query in the logs), and calculate precision and recall metrics to evaluate the CLM's performance and tune t_a. Specifically, given a query Q, we retrieve a set of cities $\{C_k|P(C_k|Q) > t_a\}$.

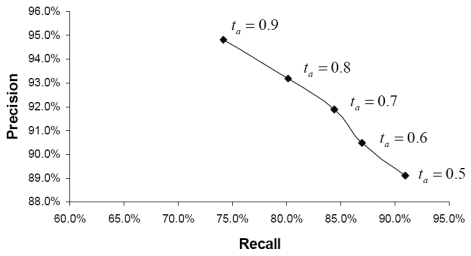


Figure 5.2. Precision/Recall curve on training subset II for location-specific query discovery.

When the ground truth city label (Q_{c_m}) is the same as the city (C_m) that has the largest value of $P(C_m|Q) > t_a$, we count that as a right decision made by the CLM in the counter N_1; but if C_m is different from its ground truth city label Q_{c_m}, we count that as a wrong decision by the CLM, using the counter N_2. We then calculate the precision P, and recall R by $P = \frac{N_1}{N_1+N_2}$ and $R = \frac{N_1+N_2}{N}$, where N denotes the number of queries in the training subset II. Intuitively, P measures the percentage of exactly right location suggestions for the suggested good location-specific queries, and R measures the percentage of suggested good location-specific queries in all the possible location-specific queries.

Figure 5.2 shows the precision/recall curve with different t_a values on the training subset II. It can be observed that by choosing $t_a = 0.7$ we can maintain reasonably high precision ($P = 92\%$) while the recall ($R = 84.4\%$) does not drop too much. We follow the same procedure to apply CLM on testing subset II where $t_a = 0.7$ achieves precision of 88%, and recall of 74%.

To further evaluate the quality of the ranked list of cities sorted by $P(C_m|Q)$, for each query (Q_{nc}) that is a location-specific query, we also compute an IR style measure called Mean Reciprocal Rank (MRR), which is the average of the reciprocal of the ranks of the correct answers to the queries in the testing data: $MRR = \sum_Q \frac{1}{r(Q)}$, where $r(Q)$ denotes the rank position of the ground truth city label (Q_c) of the location specific query, Q. The higher the MRR, the closer the correct answer's rank

position is to the top. When the correct label (Q_c) is at rank 1 for all location-specific queries (Q), the $MRR = 1$. By setting $t_a = 0.7$, we have an MRR of 0.951 on training subset II and MRR of 0.929 on testing subset II. These high MRRs imply that for location-specific queries, the true city labels appear nearly at the top of the suggested city rank list.

The above promising results, especially the high precision and MRR, show that CLMs can effectively suggest good location-specific queries and discover missing city labels. Nevertheless, our rules to discover possible location-specific queries are noisy and the automatic evaluation using (Q_c) as the ground truth city label is not very accurate. Therefore, we design human evaluation experiments to investigate the CLMs' performance by asking human editors to examine the quality of some sampled simulated location-specific queries and their city labels. Due to the high cost of human labeling, our human evaluation experiments are carried on a small set of randomly sampled queries.

5.4.3.3 Human Evaluation

We sampled a random set of queries from testing subset II, such that for each of these queries there existed at least one city, C, that was predicted such that $P(C|Q) > t_a = 0.7$, to obtain a set of 669 queries and 679 city predictions (10 queries have 2 predictions, the remaining have one). After giving a detailed explanation of the task, we asked our annotators two questions: (1) if the selected query was a location specific query and (2) if the predicted location was correct. Judges were asked to mark "Yes" or "No" in response to these questions. Eleven judges judged at least 80 predictions each and 240 predictions were judged by 2 annotators. Annotators were allowed to mark a '?' for either of the two questions. They were also allowed to use a search engine of their choice to better understand the meaning of their query. All

but two of the annotators worked in the area of information retrieval. The annotators were a mix of native and non-native speakers of English.

The inter-annotator agreement on our task was very high (84.5% on question (1) and 73% on question (2)). The disagreement on question (2) was often for ambiguous queries like "insider tv show cbs", where one annotator considered our prediction of "hollywood" as a location to be correct, since that is the location of the CBS studios. Similarly the query "city of angels tv.com" was a source of confusion, since the location in the show is Los Angeles, but the show itself is a national television show.

Of the queries that were marked location specific the accuracy of predicting a location was 84.5% [5], providing further confidence to support the rough evaluation of the previous section. However, only half of the queries of the sampled 679 were marked as location specific. Some of the error may be attributed to the explicit geo queries, obtained by using the explicit geo information analysis tool (Jones *et al.* 2008; Riise *et al.* 2003), but the remaining was due to the ad-hoc rules used for generating the data-sets used for parameter tuning. A cleaner location-specific query set or better rules may help improve the accuracy of prediction significantly. Nevertheless even this noisy data set can be used to train parameters with very high accuracy as we have seen.

5.5 Conclusion

In this chapter, we addressed an implicit information discovery challenge in web search: detecting implicit geo search intent and predicting the plausible city information related to them. Figure 5.1 (in the introduction of this chapter) depicts our approach. Our basic hypothesis is that implicit geo queries may be similar in content

[5] When we had two judgments for a query we arbitrarily selected one.

to the non-location part of explicit geo queries and that the plausible missing city level information than can be inferred from the implicit geo queries corresponds to the location part of their similar explicit geo queries.

We extracted geo language features at fine levels of granularity from large scale web search logs for this implicit information discovery challenge. We built bigram city level geo language models from web query logs so that we can calculate a query's city generation posteriors to discover its plausible geo information. In addition, we presented a rich set of geo language features through analyzing geo information units at the city level for detecting implicit geo search intent.

We used a large-scale Yahoo! web query log sample to design experiments for two implicit geo search intent analysis tasks: (1) detecting whether a query containing no explicit geo information nonetheless has geo intent and (2) discovering plausible city information in implicit geo queries. For each task, we designed a learning task for evaluating the performance, and then used minimum human-labeling effort to supervise the data and label generation to automatically obtain large-scale *synthetic* learning samples for training and testing. We leveraged click-through data as a surrogate for human labels. Experimental results demonstrated the effectiveness of our approach. In the first task of detecting users' implicit city level geo intent, the classifier achieved very high precision (more than 90%) and reasonably good accuracy (more than 74%) on the *synthetic* data. In the second task of retrieving cities in location-specific queries, the performance of city language models is very promising – CLMs achieved high precision (88%), recall (74%) and MRR (0.929) for discovering ground truth city label (Q_c) of the simulated location specific query Q. Further human evaluation results showed CLMs achieved high accuracy (84.5%) of predicting city labels for location-specific queries.

We point out that in most cases, except when creating the *testing subset I-2* (in §5.4.2) dataset, we used the non-location parts of explicit geo queries to simulate

implicit geo queries issued by users, in order to obtain large-scale evaluation data with no human labeling effort. We did not conduct the much more expensive experiment of asking human annotators to directly label *real* city-level implicit geo queries from huge volumes of queries in the web query log. Thus, we expect the performance of our approach to degrade when applying it on real data. Indeed, on the testing subset I-2 dataset obtained in a weakly supervised way in the first learning task, we observed some degradation of recall using our approach while the precision degraded very little.

There are several research directions that are worth exploring in future work. First, by human-labeling experiments in §5.4.3.3, we found that using weakly supervised *ad hoc* rules for discovering possible location-specific queries is noisy. To save human-labeling effort and also obtain more accurate location-specific query set to generate more accurate CLMs, we can explore active learning approaches (Tong and Koller 2000), through which we can select a relatively small number of samples for human judgment and automatically learn better rules to get clean location-specific query candidates. Second, we can consider building our models at a zip-code level to disambiguate between locations that have the same name. Third, we can incorporate our city language models into retrieval models for improving search performance or use the geo intent analysis results for helping to provide better query suggestions.

CHAPTER 6

CONCLUSION

In Chapter 2 to Chapter 5, we presented how we use our general perspective (Figure 1.1) to discover and use implicit information for different real-world IR challenges. We have introduced two complementary language modeling based approaches that naturally emerge from Figure 1.1 and applied them to different search tasks.

The first approach is to (a) design probabilistic generative language models to infer plausible (but missing) information for the observed queries and then (b) search plausibly relevant items that can match the original and/or the new discovered information in the queries. We discussed and evaluated this approach for each of the four IR challenges in Chapters 2 through 5.

The second approach is to use a contextual language translation approach to infer plausible implicit information for a small set of items (retrieved using the original queries) from the items' observed part. We then rerank the items based on both the original and the discovered information, in order to push the highly relevant ones up to the top of the ranked list. We used this approach for discovering anchor text and click-through information in Chapter 3 and Chapter 4, respectively. In the web search tasks in these two chapters, the discovered information of data shares vocabulary with the original textual data and is similar in nature to the unstructured queries. Thus, the discovered and the original textual information can be mixed together for better matching across varied representations of the same or similar information needs of the searchers.

6.1 Query Side vs. Searched-item Side

We first compare the complexity and the assumptions of the models used by two complementary approaches. These issues directly affect retrieval performance: for example, the model complexity affects the generalization of each approach's parameters (tuned on the training queries) for handling new queries. We discuss the efficiency of the two approaches in §6.1.2.

6.1.1 Comparisons of Model Complexity and Assumptions

Consider using the query-side and the searched-item-side (referred to as *item-side* later for simplicity) approaches for discovering anchor text. In the query-side approach (§3.4.2), we need to train 9 model parameters:

- the Dirichlet parameter μ_c; the meta-parameter β_c (in Equation 3.17), which weights the contribution of the observed field values for collecting similar web page records in the training corpus; and the meta-parameter α_c (in Equation 3.16), which weights the contribution of the extended query field for the final ranking – three parameters for the *Content* field;
- the same set of parameters μ_a, β_a and α_a for the *Associated Anchor Text* field;
- the mixture weight λ (in Equation 3.15) that controls the relative portions of the original query language model and the discovered Structured Relevance Model (SRM) to extend the query;
- the number (k) of the most similar training web page records used to build SRM for the query and the maximum number (N) of the top ranked field values in each field used for computing the final ranking score (by Equation 3.16).

In contrast, in the item-side approach (§3.4.1), we have only 3 model parameters:

- the mixture weights α and β (in Equation 3.11) that control the relative contributions of a web page's content (in an alternative view, the *Content* field),

the page's original anchor text (or the *Associated Anchor Text* field) and the inferred relevant anchor text language model (RALM) of the page to the final document likelihood ranking score of the page;

- the number (k) of the most similar training pages whose original anchor text are used for computing the RALM for a page.

From the above example, we can see that in a general case where we need to handle m different implicit data aspects for retrieval, the query-side approach needs to train $3(m+1)+3$ parameters ($m+1$ comes from m data aspects plus the original textual content of the data), while the item-side approach needs to train $2m+1$ parameters ($2m$ mixture weights and the number k described above). For example, if we want to add both the original anchor text and the auxiliary anchor text information for the task ($m = 2$), we need to train 12 parameters for the former approach but only 5 parameters for the latter. Therefore, the query-side approach is more prone to *over-fitting* than the item-side approach; i.e. the tuned parameters of the query-side approach may perform very well on the training queries while poorly on the unseen test queries. Not surprisingly, in our experiments of discovering click-through information (§4.4.3), the query-side approach (SRM) achieved the best performance on the training queries but performed worse than the web-page side approaches on the test queries in both of two ad hoc web search tasks.

Nevertheless, by using a more complex model, the query-side approach can leverage the dependencies of the field values (e.g. words or other discrete attribute values) within and across different fields for retrieval (as discussed in §2.3.2.2); thus it can effectively handle the situation where different fields (or aspects) have little overlapping or completely different vocabulary. In contrast, the item-side approach assumes that the discovered information of the data uses language similar to the data and the queries themselves, so that the language modeling based mixture models can be used during the retrieval process. When this assumption does not hold, the item-side

approach can fail. Furthermore, the query-side approach relies on the training corpus to discover implicit query field values and does not use the re-ranking scheme as the item-side approach; thus, the query-side approach can work in the situation where the straightforward approach to match query terms to the searched items fails, while the item-side approach can not. For example:

- In the resume/job record matching task (§2.5.4), running the description of a job as a query against the flattened resume collections finds hardly any matched resumes on the top-1000 positions of the returned resume ranklist, due to the large vocabulary gap between two different types of records. Thus, the mixture model approach used by the item-side approach does not work for this task because of this vocabulary gap. In addition, due to the poor quality of the top-ranked resumes, the re-ranking scheme used in the item-side approach has little hope to improve the search results. In contrast, the query-side approach (the SRM based approach) achieves reasonable performance, which brought one relevant resume to the top-5 positions of the returned ranklist, by discovering related resume words for the query job record and using them to search similar ones.

- In the NSDL record searching task (§2.5.2), because the *audience* and *subject* fields of all records in the searched collection (the test collection) are forced to be hidden, matching query fields against fields of the target records will yield no relevant results. Thus the re-ranking scheme used in the item-side approach again does not work for this task.

The query-side approach is related to typical query expansion techniques, e.g. relevance models (Lavrenko and Croft 2001), and utilizes the observed query field values for inferring plausible values for missing fields. Differently, the item-side approach is related to typical document expansion techniques, e.g. clustering-based

document smoothing techniques (Kurland and Lee 2004; Kurland and Lee 2006; Liu and Croft 2004; Tao *et al.* 2006), and utilizes the observed textual data of the searched items for inferring implicit information. Because the users' queries usually only contain very limited information – users tend to use short phrases or an incomplete sentence to describe their information need – the inferred implicit information by the query-side approach often contains irrelevant noisy values from training corpora, which can divert the search focus and degrade the search performance. This issue is known as the topic-drifting problem in typical query expansion approaches. In contrast, the item-side approach uses rich textual data of the search-items to more reliably infer implicit information, and only leverages the discovered information for smoothing; thus, this approach is more resistant to irrelevant noise from the training corpora and to the topic-drifting problem. For example:

- In the web search task of using anchor text information for finding named pages (§3.4.3), the query-side approach cannot further improve the search performance of the approach that uses each page's observed anchor text for smoothing; however, the item-side approach improves the performance statistically significantly by using discovered plausible anchor terms to smooth language models of pages.

- In the task of using click-through information discovered from web query logs for helping search the ClueWeb09 collection (§4.4.3), for the training queries, the query-side approach effectively discovers the words that may appear in those relevant pages. The discovered words mainly come from the clicked web pages of the web-query-log queries that are similar to the user query.[1] However, for the test queries, the clicked web pages of their similar web-query-log queries may

[1]Note that the tuned $\beta_q = 0.99$, which measures the relative contribution of the observed *Query Content* field to the posterior used to build the SRM for both *Query Content* and *Page Content* fields, i.e. *Page Content* missing field values are mostly inferred from the pages that have click associated-queries very similar to the user query.

contain a lot of irrelevant information, which prevents the query-side approach from gaining significant improvement of retrieval performance. In contrast, the item-side approach is less likely to be affected by irrelevant noise in the training data: it can still improve the baseline retrieval performance by using more similar pages' click-associated queries to build the RQLM for each page.

Again, we point out that both anchor text and click-associated queries in the web query logs satisfy the additional assumption of the item-side approach: the discovered information of data has partially overlapping vocabulary with the web pages and user queries so that it can be used to smooth language models of pages and bridge the vocabulary gap between the user queries and their relevant pages.

To summarize, compared with the item-side approach, the query-side approach has the following advantages: (1) it can better handle retrieval scenarios where the implicit information of the searched-items uses very different language than the original texts of those items; (2) it can find relevant items that cannot be discovered by only using the original query, by extending the query to cover every aspect of searched items for satisfying users' information need. Compared with the query-side approach, the item-side approach uses a less complex model, so that it is less prone to overfitting and performs well for unseen queries. Its retrieval performance is also more resistant to irrelevant noise in the discovered information, because (1) the searched items contain rich contextual information to infer its implicit information and (2) the discovered information is only used to smooth the original texts of the searched items during the retrieval process. Nevertheless, the item-side approach requires that the implicit data aspects share some vocabulary with the searched items and the queries; in addition, for efficiency it uses a reranking scheme which only considers the top ranked items returned by using the original query. Both issues limit the usage of the item-side approach.

6.1.2 Efficiency

Now we discuss the computational cost of the two approaches. Again, we use the example of discovering anchor text for web search.

For a given query q, the query-side approach needs to first find similar training web page records to discover missing query field values for both the *Content* and the *Associated Anchor Text* fields (denoted as field c and a, respectively). Assume that the original query contains n_q words, each word occurs $df_{avg,i}$ times on average in the field $i \in c, a$ of all the training records. It takes about $O(n_q \cdot df_{avg,c} + n_q \cdot df_{avg,a})$ time to compute the ranking scores of the returned records using the inverted list index, and $O(df_{avg,c} \cdot ln(df_{avg,c}))$ time to compute the sorted rank list – anchor text is very sparse in the collection, so we ignore $df_{avg,a}$, which is usually much smaller than $df_{avg,c}$. To compute the Structured Relevance Models (SRM) for the query from top k most similar records, we need to iterate through all the field values in each field of each returned record. Assume that the average length of the field $i \in c, a$ of the training records is $l_{avg,i}$, the cost of computing SRM is $O(k \cdot (l_{avg,c} + l_{avg,a}))$ and the cost of sorting the values in each field of the SRM to obtain the top-N most plausible field values for the second round search is $O(k \cdot (l_{avg,c} + l_{avg,a}) \cdot ln(k \cdot (l_{avg,c} + l_{avg,a})))$. Therefore, the computational cost of the first stage of the query-side approach is about:

$$\begin{aligned} C_1(q) = O(n_q \cdot (df_{avg,c} + df_{avg,a})) + O(df_{avg,c} ln(df_{avg,c})) + \\ O([1 + ln(k \cdot (l_{avg,c} + l_{avg,a}))] \cdot [(l_{avg,c} + l_{avg,a})k]), \end{aligned} \tag{6.1}$$

where $O(df_{avg,c} ln(df_{avg,c}))$ often occupies the major portion of the computing time because some words may appear in large amounts of records, which will then need to be sorted.

Similar to typical query expansion techniques, the second stage of the query-side approach – using SRM to search relevant records – is much more computationally expensive than its first stage, due to the long extended query. The union of the

inverted lists of all the values in this long query usually covers most records in the search-target collection. Therefore, assume that the search-target collection contains M_{target} records, sorting the ranking scores for these records takes $C_2 = O(M_{target} \cdot ln(M_{target}))$ time, which is the major computational cost of the second stage. As discussed in the previous section, each field of the SRM has at most N values, so there is an additional relatively small cost $O(N \cdot (df_{avg,c} + df_{avg,a}))$ of computing the ranking scores of all the records.

Note that the query-side approach needs to receive the query q first before starting all the above computation; therefore, it has high *online computational cost*. How to reduce the online computational cost of query expansion techniques while preserving their retrieval effectiveness has been recently explored (Lavrenko and Allan 2006; Cartright *et al.* 2010). Their approach (called *fast relevance models*) designs a special scheme to obtain the ranking scores of documents for the new "expanded" query without actually doing query expansion, so that the sorting cost C_2 can be greatly reduced because much less documents will be re-scored and re-ranked. Nevertheless, their special scheme needs to pre-compute the cross-entropy of all document pairs in the search-target collection before receiving queries, and thus incurs highly expensive *offline computational cost*.

Now we discuss the computational cost of the second approach – the item-side approach – using the same example of discovering anchor text for web search. This approach first retrieves a small set of web pages using the original query q. Assume that q contains n_q words and each word occurs df_{avg} times on average in search-target collection. This first stage takes $O(n_q \cdot df_{avg})$ time to use the inverted list index to compute the ranking scores for the web pages, and then takes $O(df_{avg} \cdot ln(df_{avg}))$ time to sort the pages to get the top-K ranked ones. Thus the total computational cost of this stage is $C_3(q) = O(n_q \cdot df_{avg}) + O(df_{avg} \cdot ln(df_{avg}))$.

Then the item-side approach discovers anchor text for each of the K web pages in the retrieved small set and re-ranks them. To compute the RALM for each page, we need to find its top-k most similar pages in the training collection. Assume the training collection contains M_{train} pages, when *using each page as a long query* to run again the training collection to find its most similar ones, this approach takes $C_4 = O(K \cdot [M_{train} \cdot ln(M_{train})]) + O(K \cdot [l_{avg} \cdot df_{avg}])$ time to find most similar pages for all K pages (l_{avg} is the average length of pages). The analysis of computing C_4 is similar to that of computing the cost for the second stage of the query-side approach: each of the K queries is long, so that its returned ranking score list covers almost all the M_{train} training pages and needs to be sorted in $O(M_{train} \cdot ln(M_{train}))$ time; $O(l_{avg} \cdot df_{avg})$ is the additional cost to compute the ranking score list for each page. In the end, all the RALMs can be computed in $O(K \cdot k \cdot l_{avg})$ time, and the scores of K pages can be quickly updated in $O(n_q \cdot K)$ time and re-ranked in $O(K \cdot ln(K))$ time. Thus the total computational cost of the second stage of the item-side approach is:

$$C_5(q) = C_4 + O(K \cdot k \cdot l_{avg}) + O(n_q \cdot K) + O(K \cdot ln(K)), \tag{6.2}$$

where C_4 occupies the major portion of $C_5(q)$. C_4 is *highly expensive* even for a typical setting of ad hoc search tasks where the set of pages to be reranked has the size $K = 1000$.

Note that finding a web page's similar pages and computing a RALM for the page can both be done offline (before queries are received); thus the item-side approach has highly expensive offline computational cost. The online computational cost of this approach is very small:

$$C_6(q) = C_3(q) + O(n_q \cdot K) + O(K \cdot ln(K)). \tag{6.3}$$

As shown in our previous experiments, compared with the size of the training collection (M_{train}) we only need find relatively very small ($k << M_{train}$) number of most similar pages for each page in order to compute RALM [2]. To do this, we can keep a k-element max-priority queue for each page to store its top-k most similar pages, thus avoid the expensive sorting in the offline computation. The computational cost of updating all pages' queues when adding a new page into the training collection is $O(M_{train} \cdot [ln(k) + 1])$. Recently, many advanced techniques have been proposed to address the issue of fast searching similar documents, such as the Locality-Sensitive Hashing (LSH) technique (Andoni and Indyk 2006), which uses hash techniques to map similar documents into a tight Hamming ball centered around the binary code of the query document, and the Self-Taught Semantic Hashing (STH) technique (Zhang *et al.* 2010), which uses both hash techniques and supervised learning methods to compute a more compact binary code for each document than the LSH while also mapping similar documents into similar codes as the LSH. These techniques may be used to further reduce the offline computational cost of building RALMs in the item-side approach.

To summarize, the query-side approach has high online computational cost, which mostly comes from the second round of searching that uses long extended queries. This cost depends on the size M_{target} of the search-target collection. In contrast, the item-side approach has very low online computational cost, but it has very expensive offline computational cost that mostly comes from finding top-k most similar items for each item. This offline cost depends on the size M_{train} of the training collection instead of the search-target collection, and may be greatly reduced by employing

[2] In the technique of fast relevance models (Lavrenko and Allan 2006; Cartright *et al.* 2010), the number of most similar documents of each document needed is also small in practice; therefore, the discussions here also apply for that technique.

some advanced semantic hashing techniques, which use special hash functions to map similar items to similar hash codes.

6.2 Contributions

1. We presented a general perspective for discovering plausible implicit information in large amounts of data in the context of IR (Chapter 1). This perspective leverages an intuitive assumption that data similar in some aspects are often similar in other aspects.

2. Within our general perspective, we formally developed two complementary language modeling based techniques for effectively discovering implicit information in large-scale real-world textual data for retrieval purposes: (1) the query-side approach (called Structured Relevance Models or SRM) which uses probabilistic generative models based on language models to discover implicit information for queries (§2.3); and (2) the item-side approach which builds contextual language models and employs a contextual translation approach to discover implicit information for the searched items (§3.3.2 and §4.3.2).

3. We presented how to handle empty/incomplete fields when searching semi-structured document collections, based on the query-side approach (Chapter 2).

4. Using the National Science Digital Library (NSDL) dataset, we designed experiments to empirically show that SRM (query-side) can effectively discover implicit field values in large-scale semi-structured data for synthetic missing field records (§2.4).

5. We demonstrated the effectiveness of using SRM for two real-world semi-structured records search tasks: (1) searching the NSDL collection (§2.5.2); and (2)

matching semi-structured job and resumes records in a large scale online job/resume collection (§2.5.4).

6. We presented how to handle the sparse anchor text issue for web search, using our two complementary approaches – the query-side approach (SRM) and the item-side approach (called relevant anchor text language model or RALM) (Chapter 3). The RALM technique overcomes anchor text sparsity by discovering a web page's plausible anchor text from its similar web pages' associated anchor text.

7. We designed experiments with two large-scale TREC web corpora (GOV2 and ClueWeb09) to demonstrate that RALM can effectively discover plausible anchor text for web pages with few or no in-links (§3.3). We used TREC named-page finding tasks to show that using discovered anchor text can further improve web search performance (§3.4).

8. We presented how to handle the missing/incomplete click issue when using click-through data in web query logs. We employed the query-side approach (SRM) and the item-side approach, called relevant (click-associated) query language model or RQLM, for addressing click-through sparseness (Chapter 4). We further presented how to combine RQLM with a Markov random walk approach on the click graph to further reduce click-through sparseness and improve search performance (§4.3.3).

9. Using a publicly available query log sample (Microsoft Live Search 2006 Query Log Excerpt) and two sets of TREC ad hoc web search tasks (TREC Terabyte Track 2005-2006 and Web Track 2009-2010), we demonstrated that our two approaches (SRM and RQLM) are effective (§4.4).

10. We presented how to detect web queries' underlying geo search intent and discover corresponding plausible city information, using the query-side approach (Chapter 5). We built city language models (or CLMs) for each city from the non-location part of web queries that explicitly contain the same city, and used the CLMs for implicit geo search analysis (§5.3).

11. We generated a large set of *synthetic* implicit city-level geo queries using a large-scale query log sample from the Yahoo! search engine. Then we demonstrated the effectiveness of our CLMs based approaches for predicting implicit cities for these queries (§5.4).

12. We compared the strengths and weaknesses of the query-side and the item-side approaches of discovering implicit information (§6.1). We discussed and summarized their model complexity (§6.1.1) and computational efficiency (§6.1.2).

6.3 Lessons Learned

We now summarize lessons learned from our work:

1. When implicit information of data (queries and searched items) provides helpful information for specific search tasks but is very sparse, using our discovery approaches can help to alleviate the data sparseness problem of leveraging this information for search. The data sparseness issue is common when the information is manually generated by the users, such as the user click information in the web query logs, user tag information in some collective filtering system, user-input online semi-structured forms, human-generated anchor text, etc. Discovering implicit information of data can help to reduce the semantic gap between queries and the searched-items and improve the retrieval effectiveness of IR systems. When the implicit information can help to identify searchers' specific information need or intents such as geo search intent and job-finding search in-

tent, discovering the information can help to personalize the search results and improve users' search experience.

2. When different languages are used in different implicit data aspects and/or in the original descriptions of the data, the query-side discovery approach should be used instead of the item-side approach. For example, the item-side approach does not work for the resume/job matching task because of the vocabulary gap, while the query-side approach can achieve reasonably good performance (§2.5.4).

3. When the original query fails to retrieve a large portion of relevant items (i.e. it has very low recall), the query-side approach should be used instead of the item-side approach. For example, the query-side approach works for the NSDL search task where the user-specified semi-structured query fields are completely missing in the search-target collection (§2.5.2).

4. When the original query can retrieve a reasonable number of relevant items to satisfy users' information need (i.e. it has reasonable recall), the item-side approach is a better choice because it is less prone to over-fitting and more resistant to irrelevant noise in the training collection. For example, when discovering plausible click-through queries for helping web search, the item-side approach performed well on both training and test queries (§4.4).

5. When the search task is sensitive to the topic-drifting issue and the original query can achieve reasonably good recall, the item-side approach is a better choice than the query-side approach. For example, when discovering web pages' anchor text for helping the named-page finding tasks, the item-side approach performed significantly better than other alternative approaches (§3.4).

6. Similar to typical query expansion techniques, the query-side approach has high online computational cost, which mostly depends on the size of the search-target collection and the number of query terms in the extended queries. Similar to typical document expansion techniques, the item-side approach has very low online computational cost, but it has very expensive offline cost that mostly comes from finding top-k similar items for each item (§6.1.2). It is possible that this cost can be greatly reduced by employing fast similarity search techniques such as semantic hashing techniques.

7. Our general perspective focuses on using textual similarity among data for discovering implicit information. However, when alternative information (e.g. web hyperlink graph and query-URL click graph) is available for inferring semantic relation among data, it can be combined with our approach for more accurately discovering implicit information for helping search (§3.3.1 and §4.3.1).

6.4 Future Work

In this section, we conclude the thesis by discussing three avenues of future work.

6.4.1 Combining Query Side and Searched-item Side Approaches

Previous research has shown that combining typical query expansion approach and document expansion approach can further improve the search performance, although the additional gain is very little and sometimes not statistically significant (Wei and Croft 2006; Yi and Allan 2009). We want to investigate whether combining information discovered for both the query-side and the item-side can further improve search performance.

The typical combination approach first uses document expansion techniques to get a better ranked list of documents for a given query, and then uses the top ranked documents to compute a plausibly better relevance model for query expansion and

re-retrieval (Wei and Croft 2006; Yi and Allan 2009). In this approach, the reason for doing document expansion first and then query expansion instead of the opposite way is to reduce the risk of topic-drifting from using both the expanded query and the expanded documents simultaneously to compute document ranking scores, while leveraging some advantages of both approaches. For some of our search scenarios where the original query can achieve reasonably good recall, we can follow the similar combination approach: first discover implicit information for the searched items and obtaining a better ranked list of them; then discover implicit information for queries using the top ranked items, extend the queries and perform another round of search.

However, as we discussed in §6.1.1, in some of our search scenarios, the original query may have very low recall due to the different languages in different aspects of data so that the current item-side approach is not applicable. In these situations, we need to first use the query-side approach to achieve a reasonable recall. Then we can adjust the item-side approach in the following way to rerank the top ranked items obtained by the query-side approach. That is, we do not use the mixture approach after discovering different implicit data aspects for the search items since these aspects may use very different languages. Instead, we can use the sum of the cross-entropy scores (which can be computed by Equation 2.9 in §2.5.1) between the extended queries and each extended aspect of the searched items for reranking them. Note that this approach is subject to more risk of topic-drifting. We need to do experiments to empirically evaluate whether it can achieve better retrieval performance (e.g. improving MAP or high precision region of the ranked list) by reranking.

6.4.2 Beyond "Bags of Words"

A significant amount of research has shown the great effectiveness of using word proximity information (e.g. concepts, phrases, n-grams and words that occur together in short distance in articles) in the queries and searched items for search, especially web search (Metzler and Croft 2005; Metzler and Croft 2007; Bendersky *et al.* 2009). A natural extension of our information discovery approach is to incorporate word proximity information. In Chapter 5, we showed that bigram query language models can be used to effectively analyze web searchers' fine-grained city-level geo search intent. Here, we outline how to incorporate more word proximity information into our general perspective for discovering implicit information for retrieval.

For the query-side approach, we can borrow ideas from the Markov Random Field (MRF) based Latent Concept Expansion technique (Metzler and Croft 2007) to discover plausible latent concepts that can more accurately represent users' information need for each incomplete/missing query field from the query's similar records in the training collection. Then we can use the original query and the discovered latent concepts together to search the target collection again to find more relevant records.

More formally speaking, given an m field semi-structured query $\mathbf{q} = \mathbf{q}_1 \ldots \mathbf{q}_m$ and a semi-structured record $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_m$ in the training collection $\mathcal{C}_{tn}$, assume that

1. $f_{uni}(\mathbf{q}_i, \mathbf{w}_i)$ is a unigram feature aggregation function between the ith field of $\mathbf{q}$ and $\mathbf{w}$, e.g., the sum of the log-likelihood of query terms in $\mathbf{q}_i$ appearing in the $\mathbf{w}_i$;

2. $f_{prox}(\mathbf{q}_i, \mathbf{w}_i)$ is the proximity feature aggregation function between the ith field of $\mathbf{q}$ and $\mathbf{w}$, e.g., the sum of the log-likelihood of bigrams in $\mathbf{q}_i$ appearing in the $\mathbf{w}_i$;

3. $f_{uni}(\mathbf{q}_i)$ is a query-determined unigram aggregation function, e.g. the sum of the log-likelihood of query terms in $\mathbf{q}_i$ appearing in $\mathcal{C}_{tn}$;

4. $f_{prox}(\mathbf{q}_i)$ is a query-determined proximity feature aggregation function and $f(\mathbf{w}_i)$ is a document-determined feature function, e.g. log of $\mathbf{w}_i$'s prior.

Then the MRF-based query likelihood score (Metzler and Croft 2005) between $\mathbf{q}$ and $\mathbf{w}$ can be computed by:

$$\begin{aligned} P(\mathbf{q},\mathbf{w}) = \tfrac{1}{\mathcal{Z}} \exp[\lambda_{uni} \sum_i \alpha_i f_{uni}(\mathbf{q}_i,\mathbf{w}_i) + \lambda_{prox} \sum_i \alpha_i f_{prox}(\mathbf{q}_i,\mathbf{w}_i) + \\ \lambda'_{uni} \sum_i \alpha_i f_{uni}(\mathbf{q}_i) + \lambda'_{prox} \sum_i \alpha_i f_{prox}(\mathbf{q}_i) + \lambda_{\mathcal{W}} \sum_i \alpha_i f(\mathbf{w}_i)], \end{aligned} \tag{6.4}$$

where $\mathcal{Z}$ is a normalizing constant; $\lambda_{uni}, \lambda_{prox}, \lambda'_{uni}$, λ'_{prox} and $\lambda_{\mathcal{W}}$ are weights for corresponding feature functions; α_i is the meta-parameter to control the contribution of the ith field to the likelihood; $f(\mathbf{w}_i)$ is usually set to be 0, i.e. the priors of each field in each record appears uniformly.

After using $P(\mathbf{q},\mathbf{w})$ to find a small set of similar training records ($\mathcal{R}_{\mathbf{q}}$ or pseudo-relevant records) for the query $\mathbf{q}$, we can discover a set of k plausible latent concepts $E_i = \{\mathbf{e}_{i,j} : j = 1...k\}$ for the ith field of $\mathbf{q}$ by computing the latent concept expansion likelihood (Metzler and Croft 2007):

$$P(\mathbf{e}_{i,j}|\mathbf{q}) \propto \sum_{\mathbf{w}\in\mathcal{R}_{\mathbf{q}}} P(\mathbf{q},\mathbf{w}) \exp[\lambda_{prox} f_{prox}(\mathbf{e}_{i,j},\mathbf{w}_i) + \lambda'_{prox} f_{prox}(\mathbf{e}_{i,j})], \tag{6.5}$$

and selecting the k latent concepts $E_i = \{\mathbf{e}_{i,j}\}$ that have the highest $\{P(\mathbf{e}_{i,j}|\mathbf{q})\}$. After we incorporate all the discovered E_is into the original query $\mathbf{q}$, we can use the extended query to do a second round retrieval, where the ranking scores are computed again using Equation 6.4.

For the item-side approach, we may stick to use the KL-divergence between the unigram document language models of the searched items to compute their similarity,

since those document language models can usually be accurately estimated due to the rich textual content of the searched items. Or we may add word proximity information to calculate the content similarity by viewing an item's content as a long query and then use the above Equation 6.4 to calculate query likelihood based content similarity (here the searched item only contains one field – its textual content). Then we can transform the computed similarity to a valid contextual translation probability and discover plausible latent concepts for the extra data aspect of each target item D_0. Assume that each item D_i's extra aspect is denoted as EX_i and the target item D_0's extra aspect is denoted as EX_0, similar to Equation 6.5, we can compute an expansion likelihood for each latent concept e_j given D_0[3]:

$$P(e_j|D_0) \propto \sum_{EX_i} t(EX_i, EX_0) \exp[\lambda_{prox} f_{prox}(e_j, EX_i) + \lambda'_{prox} f_{prox}(e_j)], \tag{6.6}$$

where $t(EX_i, EX_0)$ is the contextual translation probability from EX_0 to EX_i, λ_{prox}, λ'_{prox}, $f_{prox}(\cdot,\cdot)$ and $f_{prox}(\cdot)$ have similar meanings as in Equation 6.5 and 6.4. Then we can incorporate the latent concept expansion likelihood $E_i = \{P(e_j|D_i)\}$ of each item D_i into the feature functions between the query q and D_i, and then rerank items by:

$$\begin{aligned} P(q, D_i) \propto \exp[&\lambda_{uni} f_{uni}(q, D_i, E_i) + \lambda_{prox} f_{prox}(q, D_i, E_i) + \\ &\lambda'_{uni} f_{uni}(q) + \lambda'_{prox} f_{prox}(q)], \end{aligned} \tag{6.7}$$

where $f_{uni}(q, D_i, E_i)$ can be computed by the mixture model approach, i.e., each query term's generation likelihood used in this feature function is computed by the mixture of the term's original document likelihood in D_i and its latent concept expansion likelihood in E_i; and $f_{prox}(q, D_i, E_i)$ can be computed similarly by the mixture model approach.

[3]To be consistent with our previous discussions for this approach, we assume the query q and also the searched items' latent concepts are unstructured, thus we use non-bolded characters to denote them.

Therefore, for both the query-side and the item-side approaches, we can explore using the above MRF-based language modeling techniques to incorporate word proximity for more accurate implicit information discovery and more effective retrieval.

6.4.3 Beyond Language Modeling Based Retrieval

When additional implicit information of data needs to be leveraged for search, the number of the model parameters in both approaches from our general perspective will increase linearly as discussed in §6.1.1. With the linear increase of model complexity, the training cost of finding the optimal parameter setting to achieve the best retrieval performance grows exponentially when we use the brute-force way of grid-searching parameter ranges, yet the trained models becomes more prone to over-fitting. The situation will become even worse if we want to incorporate word proximity information into our general perspective as discussed in the previous section, because many more parameters have been introduced to handle latent word proximity information for each extra aspect of data.

Furthermore, there exist many IR scenarios where we want to leverage discovered non-language-modeling based information for further improving retrieval performance. For example, as mentioned in Chapter 4, we may want to use two effective click-through features (Gao *et al.* 2009): (1) the number of click-associated queries of a page in the click graph enriched by the Markov Random walk method and (2) the number of words in these queries, for improving web search. For another example, we may want to explore the utility of the smoothed non-semantic click-through features of a page (discussed at the end of Chapter 4) for web search.

To address the above issues, we consider employing machine learning techniques based retrieval approaches, known as *learning-to-rank* (Joachims 2002; Burges *et al.* 2005; Burges *et al.* 2006), to use greatly varied discovered information as a large variety of ranking features for improving search effectiveness. Generally speaking,

learning-to-rank techniques automatically learn some ranking functions[4], which can directly compute the preference order of each pair of documents or a list of documents for a given query, from training data (which include training queries and their related preference orders of document pairs or lists). This automatic learning procedure relies on optimizing certain ranking performance measurements, such as minimizing some ranking cost functions that are determined by the targeted preference order of documents in the training data. For example, in a learning-to-rank technique called RankNet (Burges *et al.* 2005), the ranking cost is a sigmoid output combined with the cross entropy cost on pairs of documents: if document i is to be ranked higher than document j, then the ranking cost is:

$$C_{i,j} = -1(s_i - s_j) + \log(1 + e^{s_i - s_j}), \tag{6.8}$$

where s_i and s_j are the scores of document i and j, respectively, output from an artificial neural network used in RankNet.

Learning-to-rank retrieval techniques can effectively incorporate intrinsically different data features into a unified machine learning process for ranking. They have been used for combining different features for different search tasks as well as finding effective ranking features for those tasks[5]. In future, we also want to use these techniques to analyze the relative utility of information discovered by different approaches for search so that we can select to discover most useful implicit information to achieve both effectiveness and efficiency.

---

[4]The complexity of the ranking function is determined by the number of features used and the model structure of the function. For example, SVM-rank (Joachims 2002) often uses thousands of features and linear/non-linear kernel function for ranking; RankNet uses thousands of features and 3 level non-linear artificial neural network for ranking (Burges *et al.* 2005). The training cost is very expensive, which involves of human-labeling cost and the offline computational cost determined by the size of training samples and the complexity of the ranking function. The online testing cost is small.

[5]The retrieval performance of these techniques also greatly relies on the quality of labeled training data besides the models and features used for ranking.

BIBLIOGRAPHY

Agichtein, E., E. Brill, and S. Dumais, 2006 Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 19–26.

Andoni, A. and P. Indyk, 2006 Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 459–468.

Andrade, L. and M. J. Silva, 2006 Relevance Ranking for Geographic IR. In *ACM workshop on Geographical information retrieval*.

Aslam, J. A. and V. Pavlu, 2007 A practical sampling strategy for efficient retrieval evaluation. Technical report.

Bendersky, M. and W. B. Croft, 2008a Discovering Key Concepts in Verbose Queries. In *Proceedings of the 31st Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–498.

Bendersky, M. and W. B. Croft, 2008b Discovering key concepts in verbose queries. In *Proceedings of ACM SIGIR*, pp. 491–498.

Bendersky, M. and W. B. Croft, 2009 Analysis of Long Queries in a Large Scale Search Log. In *Workshop on Web Search Click Data (WSCD 2009)*, pp. 8–14.

Bendersky, M., D. Metzler, and W. B. Croft, 2009 Learning Concept Importance Using a Weighted Dependence Model. In *Proccedings of Third ACM International Conference on Web Search and Data Mining (WSDM) 2010*, pp. 31–40.

Blair, D., 1988 An extended relational document retrieval model. Information Processing and Management *24*(3)**:** 349–371.

Broder, A., R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, 2000 Graph structure in the Web. Computer Networks *33*(1-6)**:** 309–320.

Broglio, J., J. P. Callan, and W. B. Croft, 1993 An Overview of the INQUERY System as Used for the TIPSTER Project. Technical report, Amherst, MA, USA.

Buneman, P., 1997 Semistructured data. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 117–121.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, 2005 Learning to rank using gradient descent. In *Proceedings of ICML*, pp. 89–96.

Burges, C. J. C., R. Ragno, and Q. V. Le, 2006 Learning to Rank with Nonsmooth Cost Functions. In *NIPS*, pp. 193–200.

Büttcher, S., C. L. A. Clarke, and I. Soboroff, 2006 The TREC 2006 Terabyte Track. In *Proceedings of TREC*.

Carterette, B., J. Allan, and R. Sitaraman, 2006 Minimal test collections for retrieval evaluation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 268–275.

Carterette, B. and R. Jones, 2007 Evaluating Search Engines by Modeling the Relationship Between Relevance and Clicks. In *Proceedings of NIPS*.

Cartright, M.-A., J. Allan, V. Lavrenko, and A. McGregor, 2010 Fast Query Expansion Using Approximations of Relevance Models. In *Proceedings of the Conference on Information and Knowledge Management*, pp. 1573–1576.

Cen, R., Y. Liu, M. Zhang, Y. Jin, and S. Ma, 2006 THUIR at TREC 2006 Terabyte Track. In *Proceedings of TREC*.

Chang, C.-C. and C.-J. Lin, 2006 *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chen, S. F. and J. Goodman, 1996 An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of ACL*, pp. 310–318.

Clarke, C. A., N. Craswell, and I. Soboroff, 2009 Overview of the TREC 2009 Web Track. In *Proceedings of the TREC Conference*.

Clarke, C. L. A., N. Craswell, and I. Soboroff, 2004 Overview of the TREC 2004 Terabyte Track. In *Proceedings of TREC*.

Clarke, C. L. A., F. Scholer, and I. Soboroff, 2005 The TREC 2005 Terabyte Track. In *Proceedings of TREC*.

Cohen, W., 2000 WHIRL: A word-based information representation language. Artificial Intelligence *118*(1–2)**:** 163–196.

Craswell, N., D. Hawking, and S. Robertson, 2001 Effective site finding using link anchor information. In *Proceedings of ACM SIGIR*, pp. 250–257.

Craswell, N. and M. Szummer, 2007 Random walks on the click graph. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 239–246.

Croft, W. B., 1995 What do People Want from Information Retrieval. D-Lib Magazine. Available at http://www.dlib.org/dlib/november95/11croft.html.

Croft, W. B., D. Metzler, and T. Strohman, 2010 Search Engines: Information Retrieval in Practice. pp. 283.

Dang, V. and W. B. Croft, 2009 Query Reformulation Using Anchor Text. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 41–50.

DeFazio, S., A. Daoud, L. A. Smith, and J. Srinivasan, 1995 Integrating IR and RDBMS Using Cooperative Indexing. In *Proceedings of SIGIR*, pp. 84–92.

Dempster, A. P., N. M. Laird, and D. B. Rubin, 1977 Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B *39*(1)**:** 1–38.

Desai, B. C., P. Goyal, and F. Sadri, 1987 Non-first normal form universal relations: an application to information retrieval systems. Information Systems *12*(1)**:** 49–55.

Diaz, F. and D. Metzler, 2006 Improving the estimation of relevance models using large external corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 154–161. ACM.

Dopichaj, P., A. Skusa, and A. Hess, 2009 Stealing Anchors to Link the Wiki. In *Advances in Focused Retrieval: INEX 2008, LNCS 5631*, pp. 343–353.

Dou, Z., R. Song, J.-Y. Nie, and J.-R. Wen, 2009 Using anchor texts with their hyperlink structure for web search. In *Proceedings of ACM SIGIR*, pp. 227–234.

Eiron, N. and K. S. McCurley, 2003 Analysis of anchor text for web search. In *Proceedings of ACM SIGIR*, pp. 459–460.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer, 1999 Learning Probabilistic Relational Models. In *Proceedings of IJCAI*, pp. 1300–1309.

Fuhr, N., 1993 A Probabilistic Relational Model for the Integration of IR and Databases. In *Proceedings of SIGIR*, pp. 309–317.

Fujii, A., 2008 Modeling anchor text and classifying queries to enhance web document retrieval. In *Proceedings of WWW*, pp. 337–346.

Gao, J., W. Yuan, X. Li, K. Deng, and J.-Y. Nie, 2009 Smoothing clickthrough data for web search ranking. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 355–362.

Geva, S., 2008 GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia. In *Proceedings of INEX 2007, LNCS 4862*, pp. 40–416.

Gey, F. C., R. R. Larson, M. Sanderson, K. Bischoff, T. Mandl, C. Womser-Hacker, D. Santos, P. Rocha, G. M. D. Nunzio, and N. Ferro, 2006 GeoCLEF 2006: The CLEF 2006 Cross-Language Geographic Information Retrieval Track Overview. In *Proceedings of CLEF*, pp. 852–876.

Gey, F. C., R. R. Larson, M. Sanderson, H. Joho, P. Clough, and V. Petras, 2005 GeoCLEF: The CLEF 2005 Cross-Language Geographic Information Retrieval Track Overview. In *Proceedings of CLEF*, pp. 908–919.

Godbole, S. and S. Sarawagi, 2004 Discriminative Methods for Multi-Labeled Classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 22–30.

Good, I., 1953 The population frequencies of species and the estimation of population parameters. Biomerika *40*(3)**:** 237–264.

Grabs, T. and H.-J. Schek, 2002 ETH Zurich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. In *Proceedings of INEX Workshop*, pp. 141–148.

Guo, J., G. Xu, H. Li, and X. Cheng, 2008 A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 379–386.

Heckerman, D., C. Meek, and D. Koller, 2004 Probabilistic Models for Relational Data. Technical Report MSR-TR-2004-30, Microsoft Research.

Hofmann, T., 1999 Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57.

Järvelin, K. and J. Kekäläinen, 2002 Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems *20*(4)**:** 422–446.

Jing, Y. and W. B. Croft, 1994 An Association Thesaurus for Information Retrieval. In *RIAO 94 Conference Proceedings*, pp. 146–160.

Joachims, T., 2002 Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142.

Jones, R., A. Hassan, and F. Diaz, 2008 Geographic features in web search retrieval. In *GIR '08: Proceeding of the 2nd international workshop on Geographic information retrieval*, pp. 57–58.

Jones, R., W. V. Zhang, B. Rey, P. Jhala, and E. Stipp, 2008 Geographic Intention and Modification in Web Search. International Journal of Geographical Information Science (IJGIS).

Kim, J., X. X. and W. B. Croft, 2009 A Probabilistic Retrieval Model for Semistructured Data. In *Proceedings of 31st European Conference on Information Retrieval*, pp. 228–239.

Koller, D. and N. Friedman, 2010 Probabilistics Graphical Models: Principles and Techniques. pp. 850–856.

Koolen, M. and J. Kamps, 2010 The importance of anchor text for ad hoc search revisited. In *Proceedings of SIGIR*, pp. 122–129.

Kurland, O. and L. Lee, 2004 Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of ACM SIGIR*, pp. 194–201.

Kurland, O. and L. Lee, 2006 Respect my authority!: HITS without hyperlinks, utilizing cluster-based language models. In *Proceedings of ACM SIGIR*, pp. 83–90.

Lafferty, J. and C. Zhai, 2001 Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *Proceedings of SIGIR*, pp. 111–119.

Lavrenko, V., 2004 A Generative Theory of Relevance. PhD dissertation, University of Massachusetts, Amherst, MA.

Lavrenko, V. and J. Allan, 2006 Real-time Query Expansion in Relevance Models. Technical Report 473, University of Massachusetts.

Lavrenko, V. and W. B. Croft, 2001 Relevance based language models. In *Proceedings of ACM SIGIR*, pp. 120–127.

Lavrenko, V., X. Yi, and J. Allan, 2007 Information Retrieval On Empty Fields. In *Proceedings of NAACL-HLT*, pp. 89–96.

Li, X., Y. Wang, and A. Acero, 2008 Learning query intent from regularized click graphs. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 339–346.

Little, R. J. A. and D. B. Rubin, 1986 *Statistical analysis with missing data*. New York, NY, USA: John Wiley & Sons, Inc.

Liu, X. and W. B. Croft, 2004 Cluster-based retrieval using language models. In *Proceedings of ACM SIGIR*, pp. 186–193.

Ma, H., I. King, and M. R. Lyu, 2007 Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 39–46.

Macleod, I., 1991 Text retrieval and the relational model. Journal of the American Society for Information Science *42*(3)**:** 155–165.

Mandl, T., P. Carvalho, G. M. D. Nunzio, F. C. Gey, R. R. Larson, D. Santos, and C. Womser-Hacker, 2008 GeoCLEF 2008: The CLEF 2008 Cross-Language Geographic Information Retrieval Track Overview. In *Proceedings of CLEF*, pp. 808–821.

Mandl, T., F. C. Gey, G. M. D. Nunzio, N. Ferro, R. R. Larson, M. Sanderson, D. Santos, C. Womser-Hacker, and X. Xie, 2007 GeoCLEF 2007: The CLEF 2007 Cross-Language Geographic Information Retrieval Track Overview. In *Proceedings of CLEF*, pp. 745–772.

Manning, C. D., P. Raghavan, and H. Schütze, 2008 Introduction to Information Retrieval. Cambridge University Press. pp. 26.

McCallum, A. K., 1999 Multi-label text classification with a mixture model trained by EM. In *AAAI 99 Workshop on Text Learning*.

Mei, Q., D. Zhang, and C. Zhai, 2008 A general optimization framework for smoothing language models on graph structures. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 611–618.

Metzler, D. and W. B. Croft, 2005 A Markov Random Field Model for Term Dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 472–479.

Metzler, D. and W. B. Croft, 2007 Latent Concept Expansion Using Markov Random Fields. In *Proceedings of the 30th Annual International ACM SIGIR Conference*, pp. 311–318.

Metzler, D., J. Novak, H. Cui, and S. Reddy, 2009 Building enriched document representations using aggregated anchor text. In *Proceedings of ACM SIGIR*, pp. 219–226.

Nallapati, R., W. B. Croft, and J. Allan, 2003 Relevant query feedback in statistical language modeling. In *Proceedings of CIKM*, pp. 560–563.

Neville, J. and D. Jensen, 2003 Collective classification with relational dependency networks. In *Proceedings of the 2nd Multi-Relational Data Mining Workshop, ACM SIGKDD*.

Neville, J., D. Jensen, L. Friedland, and M. Hay, 2003 Learning relational probability trees. In *Proceedings of ACM SIGKDD*, pp. 625–630.

Neville, J., D. Jensen, and B. Gallagher, 2003 Simple Estimators for Relational Bayesian Classifiers. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, Washington, DC, USA, pp. 609. IEEE Computer Society.

Ogilvie, P. and J. Callan, 2003 Combining document representations for known-item search. In *Proceedings of ACM SIGIR*, pp. 143–150.

Pasca, M., 2007 Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM*, pp. 683–690.

Ponte, J. M. and W. B. Croft, 1998 A language modeling approach to information retrieval. In *Proceedings of ACM SIGIR*, pp. 275–281.

Purves, R. and C. Jones, 2007 GIR '07: Proceedings of the 4th ACM workshop on Geographical information retrieval. New York, NY, USA. ACM.

Radlinski, F. and T. Joachims, 2007 Active exploration for learning rankings from clickthrough data. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 570–579.

Raghavan, H., J. Allan, and A. McCallum, 2004 An Exploration of Entity Models, Collective Classification and Relation Description. In *Proceedings of ACM LinkKDD*, pp. 1–10.

Riise, S., D. Patel, and E. Stipp, 2003 Geographical Location Extraction. US Patent Application 20050108213.

Robertson, S. E., 1991, January)On term selection for query expansion. Journal of Documentation **46:** 359–364.

Rocchio, J. J., 1971 Relevance feedback in information retrieval. In: G. Salton (ed.), The SMART Retrieval System C Experiments in Automatic Document Processing**:** 312–323.

Rousu, J., C. Saunders, S. Szedmak, and J. Shawe-Taylor, 2006 Kernel-Based Learning of Hierarchical Multilabel Classification Models. Journal of Machine Learning Research **7:** 1601–1626.

Sanderson, M. and J. Kohler, 2004 Analyzing geographic queries. In *ACM workshop on Geographical information retrieval*, Sheffield, UK.

Seo, J., W. B. Croft, K. Kim, and J. Lee, 2011 Smoothing Click Counts for Aggregated Vertical Search. In *Proceedings of 33st European Conference on Information Retrieval, to appear.*

Silverman, B., 1986 *Density Estimation for Statistics and Data Analysis*, pp. 75–94. CRC Press.

Tang, L., S. Rajan, and V. K. Narayanan, 2009 Large scale multi-label classification via metalabeler. In *WWW '09: Proceedings of the 18th international conference on World wide web*, New York, NY, USA, pp. 211–220. ACM.

Tao, T., X. Wang, Q. Mei, and C. Zhai, 2006 Language model information retrieval with document expansion. In *Proceedings of NAACL-HLT*, pp. 407–414.

Taskar, B., P. Abbeel, and D. Koller, 2002 Discriminative probabilistic models for relational data. In *Proceedings of UAI*, pp. 485–492.

Taskar, B., E. Segal, and D. Koller, 2001 Probabilistic classification and clustering in relational data. In *Proceedings of IJCAI*, pp. 870–876.

Tong, S. and D. Koller, 2000 Support Vector Machine Active Learning with Applications to Text Classification. In *Proceedings of ICML*, pp. 999–1006.

Van Rijsbergen, C. J., 1979 Information Retrieval.

Vasanthakumar, S. R., J. P. Callan, and W. B. Croft, 1996 Integrating INQUERY with an RDBMS to Support Text Retrieval. IEEE Data Engineering Bulletin *19*(1)**:** 24–33.

Vassilvitskii, S. and E. Brill, 2006 Using web-graph distance for relevance feedback in web search. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 147–153.

Vries, A. and A. Wilschut, 1999 On The Integration of IR and Databases. In *Proceedings of IFIP 2.6 Working Conference on Data Semantics*, Rotorua, New Zealand.

Wang, L., C. Wang, X. Xie, J. Forman, Y. Lu, W.-Y. Ma, and Y. Li, 2005 Detecting dominant locations from search queries. In *Proceedings of ACM SIGIR*, pp. 424–431.

Wang, X. and C. Zhai, 2008 Mining term association patterns from search logs for effective query reformulation. In *Proceeding of CIKM*, pp. 479–488.

Wei, X. and W. B. Croft, 2006 LDA Based Document Models for Ad hoc Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pp. 178–185.

Welch, M. J. and J. Cho, 2008 Automatically Identifying Localizable Queries. In *Proceedings of ACM SIGIR*, pp. 507–514.

Westerveld, T., W. Kraaij, and D. Hiemstra, 2001 Retrieving Web Pages using Content, Links, URLs and Anchors. In *Proceedings of the TREC Conference*, pp. 663–672.

Xu, J. and W. B. Croft, 1996 Query Expansion Using Local and Global DocumentAnalysis. In *Proceedings of the 19th International Conference onResearch and Development Information Retrieval (SIGIR96)*, pp. 4–11.

Xue, G.-R., H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan, 2004 Optimizing web search using web click-through data. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pp. 118–126.

Yi, X. and J. Allan, 2009 A Comparative Study of Utilizing Topic Models for Information Retrieval. In *Proceedings of 31st European Conference on Information Retrieval*, pp. 29–41.

Yi, X. and J. Allan, 2010 A Content based Approach for Discovering Missing Anchor Text for Web Search. In *Proceedings of the 33rd Annual ACM SIGIR Conference*, pp. 427–434.

Yi, X., J. Allan, and W. B. Croft, 2007 Matching resumes and jobs based on relevance models. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 809–810.

Yi, X., J. Allan, and V. Lavrenko, 2007 Discovering Missing Values in Semi-Structured Databases. In *Proceedings of RIAO 2007 - 8th Conference - Large-Scale Semantic Access to Content (Text, Image, Video and Sound).*

Yi, X., H. Raghavan, and C. Leggetter, 2009 Discovering users' specific geo intention in web search. In *WWW '09: Proceedings of the 18th international conference on World wide web*, New York, NY, USA, pp. 481–490. ACM.

Yu, B. and G. Cai, 2007 A query-aware document ranking method for geographic information retrieval. In *ACM workshop on Geographical information retrieval*, pp. 49–54.

Zhai, C. and J. Lafferty, 2001a Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, pp. 403–410.

Zhai, C. and J. Lafferty, 2001b A Study of Smoothing Methods for Language Models Applied to Ad-Hoc Information Retrieval. In *Proceedings of ACM SIGIR*, pp. 334–342.

Zhang, D., J. Wang, D. Cai, and J. Lu, 2010 Self-taught hashing for fast similarity search. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 18–25.

Zhu, S., X. Ji, W. Xu, and Y. Gong, 2005 Multi-labelled classification using maximum entropy method. In *Proceedings of ACM SIGIR*, pp. 274–281.