RETRIEVAL AND EVALUATION TECHNIQUES FOR PERSONAL INFORMATION

A Dissertation Presented

by

JINYOUNG KIM

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 2012

Department of Computer Science

RETRIEVAL AND EVALUATION TECHNIQUES FOR PERSONAL INFORMATION

A Dissertation Presented

by

JINYOUNG KIM

ABSTRACT

RETRIEVAL AND EVALUATION TECHNIQUES FOR PERSONAL INFORMATION

SEPTEMBER 2012

JINYOUNG KIM

B.Sc., SEOUL NATIONAL UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

Providing an effective mechanism for personal information retrieval is important for many applications, and requires different techniques than have been developed for general web search. This thesis focuses on developing retrieval models and representations for personal search, and on designing evaluation frameworks that can be used to demonstrate retrieval effectiveness in a personal environment.

From the retrieval model perspective, personal information can be viewed as a collection of multiple document types each of which has unique metadata. Based on this perspective, we propose a retrieval model that exploits document metadata and multi-type structure. Proposed retrieval models were found to be effective in other structured document collections, such as movies and job descriptions.

Associative browsing is another search method that can complement keyword search. To support this type of search, we propose a method for building an association graph representation by combining multiple similarity measures based on a user's click patterns. We also present a learning techniques for refining the graph structure based on user's clicks.

Evaluating these methods is particularly challenging for personal information due to privacy issues. This thesis introduces a set of techniques that enables realistic and repeatable evaluation of techniques for personal information retrieval. In particular, we describe techniques for simulating test collections and show that game-based user studies can collect more realistic usage data with relatively small cost.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

According to Jones and Teevan [42], Personal Information Management (PIM) refers to both the practice and study of the activities a person performs in order to locate or create, store, organize, maintain, modify, retrieve, use and distribute information in each of its many forms as needed to meet many goals and to fulfill life's many roles and responsibilities. As is clear in this definition, the value of PIM activities can be found in the context of achieving other goals in lives, and Jones [41] expressed this more succinctly that *PIM is the art of getting things done in our lives through information.*

A central challenge in PIM is the problem of retrieving (or accessing) one's information, and this is the focus of this thesis: designing and evaluating techniques for personal information retrieval (PIR). Although personal information can take various forms, we define PIR as the process involved in a person accessing their own information stored in digital form. This includes accessing documents in one's desktop or mobile devices, as well as personal information in the various types of social media.

This is a significant issue because the amount and variety of information we deal with in our everyday lives is constantly growing and current search tools are inadequate. Moreover, it is known that effective and efficient information access is a critical component for the productivity of knowledge workers. While web search has changed how people access information on the web, and personal information is increasingly spread across various web services, finding information in one's own digital collection remains a difficult task for most people.

Despite the importance of the task, research in PIR has been relatively stagnant for several reasons. First, since each person has a different mix of information, created using a variety of tools, it is hard to design a retrieval approach that generalizes across all users. Second, evaluating PIR has been considered costly, because it typically involves long-term user studies where participants are expected to use the software provided during the period of the experiment. Finally, the data collected during such user studies cannot be shared with other researchers due to privacy concerns.

This thesis aims to develop and evaluate a set of techniques that enables effective retrieval of personal information. To avoid the aforementioned problems, we take several new approaches. We propose general retrieval models that are applicable regardless of the characteristics of a user's data or behavior—a field-based search model and an associative browsing model. We also introduce evaluation methods—Pseudo-desktop and DocTrack—that make it possible for any PIR system to be evaluated without a long-term user study, and for the outcome of the evaluation to be used by other researchers. Both the retrieval models and the evaluation methods are extensively verified in a variety of settings.

Since these are general techniques motivated by several characteristics of personal information, they have applications beyond PIR. For instance, the field-based retrieval framework can be used for arbitrary collections with rich structural metadata. Especially, we find that the proposed retrieval model (the field relevance model) outperforms strong baselines in several structured document collections. Also, the evaluation techniques presented here are applicable to other areas such as enterprise search which also deals with privacy-sensitive data.

In the remainder of this chapter, we first give an overview of the problem domain and present several key observations. Based on these observations, we then introduce the approaches taken by this thesis in detail, followed by the research contributions to be made.

1.1 Personal Information Retrieval

In this section, we first define and characterize the problem in detail. We then describe several recent trends, followed by the discussion of the problem characteristics on which we build our approaches.

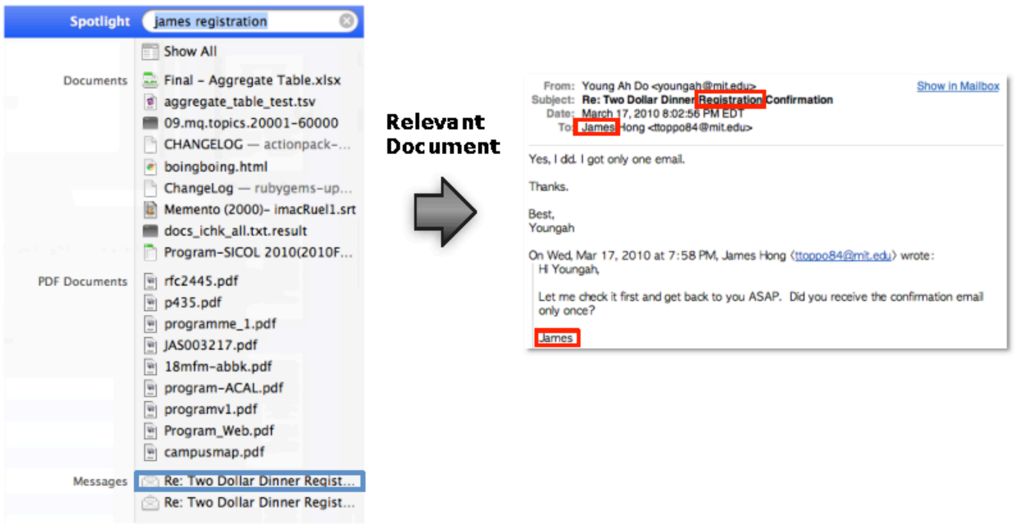
1.1.1 Problem Definition

We first provide a more broad definition of the problem of personal information retrieval (PIR), and specify where we focus on in this thesis. This is an important step because there are many different types of personal information, user's information needs for personal information, and possible solutions for addressing them.

There exist two major forms of personal information: analog (mostly paper-based) and digital. Since a greater portion of personal information is moving into digital form (e.g., the availability for easy scanning solution for personal documents), and the information in digital form is more amenable to automated retrieval techniques, we focus on digital information.

In terms of user's information needs, the nature of personal information indicates that most of information objects are the ones seen or created by the user. This in turn means that most of target items for retrieval are the ones seen by the user, where users can use their memory during the retrieval process. Although accessing personal information can involve other types of information needs, we focus on this so-called known-item finding problem, which is known as the most frequent form of information needs [34] [42] in PIR.

Finally, the problem of PIR can be approached both from human and technological perspective, since effective and efficient information access is a function of both the user and the system. Solutions from human perspective involves educating the user on how to keep the information so that retrieval is made easier (e.g., foldering or tagging), or how to find things more efficiently using the retrieval facilities of the



```
james registration

Final – Aggregate Table.xlsx
aggregate_table_test.tsv
09.mq.topics.20001-60000
CHANGELOG — actionpack-...
boingboing.html
ChangeLog — rubygems-up...
Memento (2000)- imacRuel1.srt
docs_ichk_all.txt.result
Program-SICOL 2010(2010F...
rfc2445.pdf
p435.pdf
programme_1.pdf
JAS003217.pdf
18mfm-abbk.pdf
program-ACAL.pdf
programv1.pdf
Program_Web.pdf
campusmap.pdf
Re: Two Dollar Dinner Regist...
Re: Two Dollar Dinner Regist...
```

```
From:    Young Ah Do <youngah@mit.edu>
Subject: Re: Two Dollar Dinner Registration Confirmation
Date:    March 17, 2010 8:02:56 PM EDT
To:      James Hong <ttoppo84@mit.edu>

Yes, I did. I got only one email.

Thanks.

Best,
Youngah

On Wed, Mar 17, 2010 at 7:58 PM, James Hong <ttoppo84@mit.edu> wrote:
 Hi Youngah,

 Let me check it first and get back to you ASAP.  Did you receive the confirmation email
 only once?

 James
```

Figure 1.1. A typical scenario of personal information retrieval: searching for a known-item in a desktop. In the ranked list of retrieval results (left), the target document is highlighted with a box. The relevant document (right) has matches with the query in many different parts

system. From the technological perspective, one can develop a complete system that helps the user access personal information, or develop techniques which can be a part of such systems. Here we focus on developing techniques that are generalizable beyond the problem domain we are dealing with.

In order to discuss the problem in a more concrete manner, consider the following scenario. A user is looking for an email regarding event registration from a person whose first name is James. Based on what she remembers about the email, she types in a query ('james registration') on her computer. She gets the results shown in Figure 1.1, which contains the email she had in mind near the bottom of the ranked list. Although there would be many other scenarios of PIR, this case exemplifies the most common scenario: the search for a known item on the user's desktop.

1.1.2 Problem Characteristics

In order to develop a sensible approach to the problem, here we intend to characterize the problem of PIR in terms of the document collection, user behavior and research methodology.

From the perspective of the data, we can regard personal information as a collection of multiple document types with type-specific metadata. For instance, emails have *sender* and *receiver* fields, whereas office documents have *filename* and *author* fields. Considering that personal information is now increasingly spread across various places on the web (e.g., blog, Twitter, Facebook), this characterization is even more valid.

From the perspective of the task, we primarily focus on the known-item search for this thesis. The known-item search task implies that people rely on their memory of target documents during the information-seeking process, and we use this property extensively in the design of retrieval and evaluation techniques. For instance, the assumption that the user has a specific target document in mind turns out to become critical in the design of the simulated evaluation technique we propose.

Another important point is that a PIR system continually interacts with a single user over a long period, unlike a web search engine which serves the sporadic information needs of many individuals. This long-term interaction provides opportunities for the system to better understand and serve the user, where the challenge lies in adapting to different behavioral patterns of different users. In this thesis, we suggest several techniques by which the behavior of system can be personalized based on interactions with the user.

From the perspective of evaluation, the study of PIR is considered hard due to the privacy of data being searched. Unlike other areas of information retrieval for which we can build standard collections that can be shared with many other researchers, the access of documents and log data for PIR research is typically confined to the group of researchers who actually performs the study. Since this has been a major barrier for the progress of PIR research community as a whole, we suggest several evaluation methods that address this concern.

1.1.3 Recent Trends

Unlike its analog counterpart, the access of personal information has been changing significantly over the years due to the change in the hardware and software platforms and applications. Since it is important to check the validity of the proposed solutions in the context of these changes, we discuss several recent trends and whether the characteristics introduced above holds true.

First, personal information is increasingly stored and accessed in various social media in recent years. The fragmented nature of these archives present challenges in accessing such types of personal information. However, we argue that the approaches proposed here are applicable to these scenarios, since they increase the diversity of the collections.

The screenshot in Figure 1.2 from a popular web service Greplin [1] exemplifies a search in this scenario, where user's search for 'social media' matches with items from various social media streams. Comparing with the search in the desktop shown in Figure 1.1, it is clear that the underlying framework is the same, although the type of information indexed would be different.

Second, Google started a serviced called 'Search Plus You'[2] where personal information is blended into a typical Google web search results. This further accelerates the trend of integrating personal and other sources of information, where way users can see results collected from both personal and public sources. Even here, the same

[1]http://www.greplin.com

[2]http://www.google.com/insidesearch/features/plus/index.html

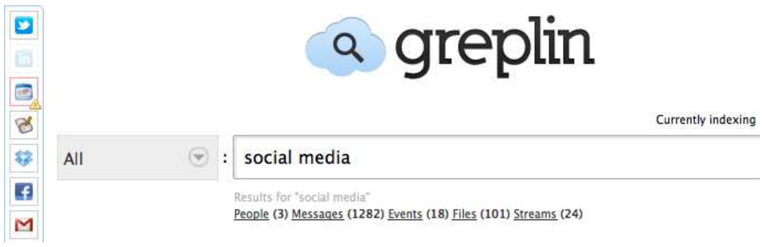
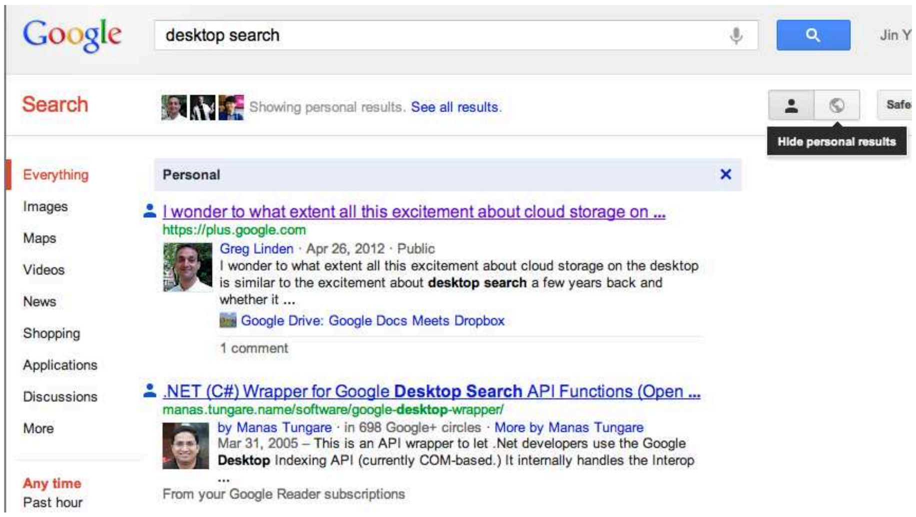

greplin
Currently indexing
All
social media
Results for "social media"
People (3) Messages (1282) Events (18) Files (101) Streams (24)

Figure 1.2. A search interface over personal social media collections.



Showing personal results. See all results.

Personal

I wonder to what extent all this excitement about cloud storage on ...
https://plus.google.com
Greg Linden · Apr 26, 2012 · Public
I wonder to what extent all this excitement about cloud storage on the desktop is similar to the excitement about **desktop search** a few years back and whether it ...
Google Drive: Google Docs Meets Dropbox
1 comment

.NET (C#) Wrapper for Google **Desktop Search** API Functions (Open ...
manas.tungare.name/software/google-**desktop**-wrapper/
by Manas Tungare · in 698 Google+ circles · More by Manas Tungare
Mar 31, 2005 – This is an API wrapper to let .Net developers use the Google **Desktop** Indexing API (currently COM-based.) It internally handles the Interop ...
From your Google Reader subscriptions

Figure 1.3. A search interface which shows the combination of personal and web collections.

framework is applicable if the web is considered as an additional source from which the results are collected.

Finally, Apple's Siri voice agent technology[3] allows smartphone users to access various types of information from personal sources and the web. By providing a more natural interaction method, Siri can provide information even when users are not able to type queries and read the results. The more complex queries that can result from this interaction can be exploited in the retrieval models we develop.

[3]http://www.apple.com/iphone/features/siri.html

1.2 Our Approach

In this section, we introduce our approach to the problem. We first describe an abstract model of information seeking in personal information retrieval that acts as a foundation on which we build our approach. We then introduce the proposed retrieval models and evaluation techniques in detail.

1.2.1 Model of Information Seeking

Given our goal of improving information access over personal document collection, it would be helpful to build an abstract model of the problem from which specific approaches can be derived. Figure 1.4 shows a model of information seeking over personal documents. Here, the user has some information need over a target document, or a set of them. For instance, she might need a email to look up a specific number.

Since we assume that she has seen the document before, she has a partial memory of the items, which allows her to formulate queries. There can be several different types of memory known in literature [84] [28]. For a typical case, if a user recalls terms from the target document, those terms can be used to formulate keyword queries. Alternatively, one may remember about documents related to the target document. For instance, the user can recall another email in the same thread (if the target document was an email), or a document one was editing at the same time with the target document.

From retrieval perspective, the user's information need somehow needs to be translated into a concrete query to initiate the retrieval process, and it is important for a system to support diverse kinds of queries that correspond to various information-seeking scenarios. For this reason, we designed our retrieval framework so that it can supports both keyword and item-based queries, which are based on the memory

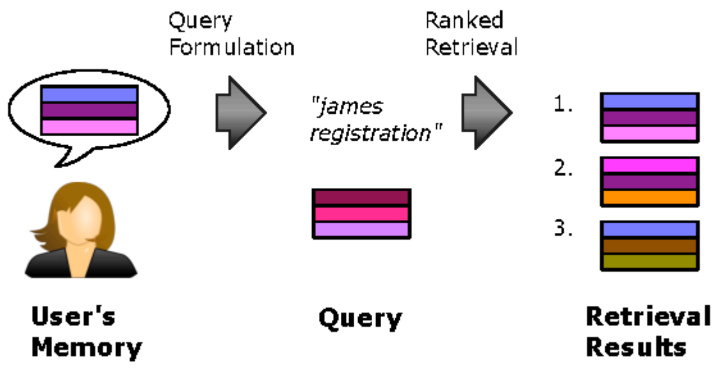


Figure 1.4. A model of information seeking in personal information retrieval.

of terms or related items, respectively. This means that users can find the target document regardless of the type of memory one has.

Finally, these queries will be used as a input for the retrieval system to produce results in the form of ranked list. Note that items have structures in the form of metadata fields, although we assume that users would type in keyword queries without any structure. The overall process from the query formulation to the ranked retrieval forms a framework on which we establish our approaches.

Specifically, our retrieval framework concerns the finding of relevant results with respect to a user's query. A major part of our evaluation framework builds upon a model of the user's information seeking process in general and query formulation in particular, with a goal of creating a realistic simulation of the procedure.

1.2.2 Retrieval Framework

Based on the model of information seeking described above, we propose using search and associative browsing as the primary retrieval models for personal information. More specifically, search provides a capability for a user to find documents across all one's digital collections using a keyword query. Since we use the structured

Table 1.1. Comparison between term-based search and associative browsing

	Field-based Search	Associative Browsing
User's Knowledge	Target document	Related document(s)
User's Input	Type a query	Click on a suggestion

nature of the personal information extensively to improve the search capability, we call it a field-based search.

Associative browsing enables the user to browse through documents by following the chain of associations, where one item functions as a query to retrieve relevant items. By providing a way to utilize the user's memory of associations between documents, associative browsing can provide an alternative mode of information access. Table 1.1 compares these two retrieval models, which shows that these methods are based on different assumptions around user's knowledge, and requires different types of input from the user.

The proposed search and browsing models are general techniques of retrieval which are applicable regardless of document types, unlike some access mechanisms that are available only to specific document types (e.g., the hierarchy of a file organization, the tags of blog posts). Also, these two methods are complementary in that they can be naturally combined in a single system. For instance, users can initiate retrieval using a keyword search, subsequently relying on associative browsing in case the keyword search is unsuccessful.

Here, we provide a concrete example on how keyword search and associative browsing can be combined for a known-item finding task. Imagine a user who is trying to find a webpage she has seen before. Further assume that she cannot come up with a good keyword for search, yet she remembers the sender of a related email. Using our approach, as shown in Figure 1.5, the user can first search for a relevant email using the person's name as a keyword query, and then browse into the target document (webpage).

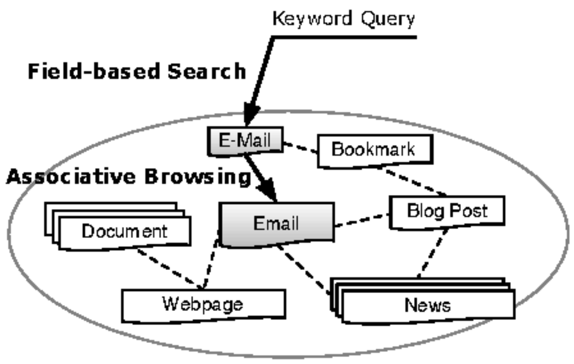


Figure 1.5. An illustration of how keyword search and associative browsing can be combined. Dotted lines represent the associations between documents and concepts. Directed lines denote how a user can access the target webpage by using term-based search first, and then associative browsing.

In developing these retrieval methods, we have several goals in mind. First of all, we aim to minimize users' effort. For field-based search, this is accomplished by predicting the type of documents users are looking for, so that a user can find all types of documents using a single search box. For associative browsing, we propose a technique for automatically building the associations between items without relying on a users' annotations.

Secondly, we build adaptive methods that can use natural interactions with the user to improve effectiveness without conscious effort. Our field-based search model employs the user's choice of relevant document to refine the estimation of field weights, or uses query logs to improve the performance of type prediction. The associative browsing model exploits click feedback to refine suggestions for browsing.

1.2.3 Evaluation Framework

There are several challenges in evaluating PIR in general, and the proposed retrieval framework in particular. First, the evaluation of any technique for searching personal information naturally involves the use of personal data, which raises privacy concerns. Second, the evaluation should take into account the diversity in a user's personal information, information needs and behavioral characteristics. Finally, since the proposed retrieval framework involves the combination of multiple information seeking methods, the evaluation method should be capable of handling such complexity in interaction methods.

Previous studies of PIR often involved an instrumentation-based user study—deploying the system in a real environment and having it evaluated by actual users. This kind of evaluation method has its own benefits, and would be required if external validity is of the utmost importance. However, it requires considerable resources, which makes it hard, if not impossible, to perform a large-scale user study. Moreover, the collections and usage logs from these studies are not open to other researchers because they include private information.

An alternative evaluation paradigm is the batch evaluation using test collections, known as Cranfield method and popularized by the TREC [4] conference in the information retrieval community. Typically composed of the document collection and a set of query and relevant documents pairs, a test collection enables a set of retrieval methods to be evaluated against one another in a repeatable manner based on a set of metrics. While this kind of batch evaluation is adopted as a standard in most of IR tasks, the private nature of personal information has prevented such test collections from being built.

In evaluating our retrieval methods, we propose a set of techniques that addresses these issues and meet the challenges in PIR evaluation. The main idea is to simulate a component of existing evaluation techniques in a way that overcomes the limitations while preserving the validity of evaluation. The target of simulation includes collection

[4] http://trec.nist.gov

documents, search tasks and even the user's interactions with the system. In what follows, we introduce the techniques in greater detail.

First of all, we propose a technique for gathering collection documents and simulating users' interactions with the system, thereby requiring no human involvement. We first employ various techniques to build document collections with similar characteristics to personal information archives. For evaluating term-based search, we developed a method for automatically generating query and target document pairs (Pseudo-desktop). For evaluating associative browsing, we propose a method for generating click behavior for browsing.

We also propose a methodology for game-based user studies where we develop simulated search tasks, and provide participants with an environment for accomplishing these in a competitive environment. We developed a human computation game (DocTrack) whose goal is to find a target document by combining the search and browsing facilities provided. Since we use public documents for such experiments, the data gathered from the game has the additional benefit of being free from privacy concerns, opening possibilities for the findings to be validated by other researchers.

1.3 Contributions

This thesis has made the following contributions, which resulted in seven publications (six papers and one poster). In what follows, we listed major contributions along with corresponding references.

- A novel field-based retrieval method for structured documents called PRM-S (Probabilistic Retrieval Model for Semi-structured data), which exploits the implicit mapping between query-terms and document fields. The PRM-S is shown to outperform existing retrieval methods for structured documents significantly in the IMDB movie collection, the Monster résumé collection and the TREC email collection. [51]

- The notion of field relevance as the generalization of per-term field weighting in PRM-S, and the corresponding retrieval method called FRM (Field Relevance Model). The FRM is shown to outperform PRM-S in the structured document collections mentioned above. [48]

- A set of techniques called Pseudo-desktop for building simulated test collections aimed at evaluating personal information retrieval. Specific contributions include novel methods for generating and validating queries for known-item search. [46]

- A method for performing game-based user studies for personal information retrieval called DocTrack. We built the CS collection by collecting public documents in UMass Computer Science department and gathering a large quantity of known-item queries and users interactions in this environment. [47]

- A probabilistic model of the user behavior for the simulated evaluation of known-item finding. The model is capable of generating the user's interaction with the system for both search and browsing, and the aggregated outcome of the model gives comparable results with user studies. [50] [49]

- A novel type prediction method for the multiple collections of structured documents called FQL (Field-based collection Query-Likelihood), which uses field-level evidences in collection scoring. The FQL method is shown to have higher accuracy than existing methods in both the Pseudo-desktop connections and the CS collection. [47]

- An adaptive type prediction method for personal information retrieval, which combines many existing type prediction methods as features to improve performance further. The suggested method is shown to have higher accuracy in the CS collection. [47]

- An adaptive method for suggesting associations between documents or concepts (the entities and terms of interest to the user) based on the user's click feedback. The suggested method is shown to be more effective in suggesting associations between items than existing methods using the CS collection. [45]

1.4 Organization

This thesis is organized as follows: in Chapter 2, we overview related work in the areas of structured document retrieval, personal information management and desktop search, and others. In Chapter 3, we describe the field-based search model, focusing on field-based retrieval models and type prediction methods. In Chapter and 4, the associative browsing model and corresponding learning framework is introduced.

In Chapter 5, we change our focus to the evaluation techniques and describe simulation-based evaluation methods in detail. In Chapter 6, we describe results for the term-based search model and associative browsing model using the proposed evaluation methods. We decided to put the experimental results at the end, since it relies on the understanding of proposed evaluation methods as well as the retrieval methods. Finally, we conclude this thesis in Chapter 7, describing future work as well as summarizing the main points.

CHAPTER 2
RELATED WORK

In this chapter, we describe related work in many areas, and how our research relates to and extends this work. In terms of problem domain, this work belongs to the category of personal information management, especially the retrieval of personal information. From the technical standpoint, the field-based search model extends techniques in known-item search, semi-structured document retrieval, and federated search. The proposed browsing model extends the research in associated browsing, and the evaluation framework is related to the field of simulated evaluation for information retrieval.

2.1 Personal Information Management

As briefly introduced at the beginning of this thesis, the field of personal information management (PIM) deals with the general problem of keeping, finding and managing personal information in general. We discuss several points by which the PIM research relates to this thesis.

Although the focus of this thesis is in providing access to personal information, the activity of accessing is intertwined with other activities in user's practice of PIM. In particular, Jones and Teevan [42] [68] point out that the user's keeping or managing activities can later affect the difficulty or the model of finding information. For instance, users can save their files in a specific location, or create annotations (e.g., tags) so that those items can be found more easily later on. The retrieval techniques proposed in this thesis, by providing an effective means of utilizing these user-created metadata for retrieval, can benefit from such user activities.

It has been recognized [80] that people use multiple modes of interaction to access their information, based on their level of knowledge and behavioral characteristics. Recently, the relationship between keyword search and other access methods (e.g., browsing through folders) has been investigated [68] [10]. In both studies, performed based on a traditional metadata-based file search system and a full-text search system, the authors conclude that people use keyword search capability rather infrequently (4-15%). Our work acknowledges the fact that keyword search is not always the preferred mode of information access, and propose the combination of search and browsing as the overall retrieval framework.

The landscape of personal information management is continuously changing [41]. Personal information is increasingly scattered across many devices, applications and online services, especially due to the widespread use of mobile devices and 'apps'. Moreover, the introduction of new hardwares and softwares, or even new versions of existing ones, require adaptation by users. This rapid turnover of the practice of PIM has motivated us to focus on the enabling technology as opposed to focusing on a particular application.

2.2 Personal Information Retrieval

In this section, we introduce related work specifically focusing on the retrieval side of personal information. This includes studies on traditional desktop search, as well as recent works on personal metasearch (personal information beyond the desktop). We also discuss several attempts to evaluate retrieval techniques over personal information.

Desktop search systems such as Stuff I've Seen [30] and Phlat [27] showed that user interaction is a significant issue in the desktop environment, and that the *date* can be the most important ranking feature since most users sorted the results by the date. Other researchers focused more on improving the quality of ranking and showed that temporal locality and causality [74] are useful features. Learning feature weights with training data [21] has also been found to be effective in the desktop environment.

From the evaluation standpoint, all these studies evaluated their approach based on a large deployment of their software within their own organization. While having the system used by actual users is certainly valuable, the private nature of collected data prevented them from being shared, thereby eliminating the possibility of comparative evaluation. Also, the procedure of system building and deployment can be quite time-consuming. Our focus is more on the underlying technology that enables effective retrieval than system building, and we introduce several evaluation methods that overcome some of these limitations.

There have been attempts to define a framework for desktop search. Thomas et al. [83] regarded desktop search as a meta-search problem where the results from many servers are merged. The work in Chapter 3 employs a similar overall framework for keyword search, and provides end-to-end evaluation of the retrieval framework. They also employed an evaluation technique [81] [82] based on the side-by-side comparison of two search results, finding that it is more effective in capturing the context of user's search. We believe our technique of constructing test collections can provide complementary evidence to theirs, and we found reasonable agreement among two methods.

The evaluation of desktop search or, in general, personal information retrieval (PIR), has been considered a challenging problem [42] because real desktop collections are not available for research due to privacy concerns. The performance evaluation of major commercial desktop search engines was tried [58] in standard IR evaluation settings, using TREC Robust track data. Chernov et al. [18] [17] proposed a method for creating a testbed for desktop search by collecting documents and queries collaboratively, yet no experimental validation was done. Elsweiler [34] suggested an evaluation method for PIM based on user studies. The approach described in Chapter 5 is different in that it does not require any direct user involvement. [33]

2.3 Known-item Search

Since the focus of this thesis is on improving access for user's personal information, and it is reasonable to assume that users have some knowledge of the documents stored in their personal files, the research in known-item search problem has particular significance to this work. The TREC 2005 Enterprise Track [23] provided a known-item email retrieval task, where a set of emails and corresponding queries were given. Among the participants, the BM25F model [22] combined a variety of document fields and other features such as the year and the thread structure to get good effectiveness. Another approach [89] combined different independent sources to improve the performance of known-item search. The PRM-S retrieval model in Section 3.2.3 outperformed the methods described above in a recent evaluation [46].

A related but different concept is re-finding, which means accessing the information previously sought for by the same user. As there are other ways of knowing an item other than finding it, re-finding can be considered a subproblem of known-item. For instance, if a user is trying to find a document that one has downloaded from the web, it is a known-item search, but not re-finding. Teevan et al. [79] [1] analyzed the re-finding behavior on the web, concluding that 40% of web queries are re-finding queries. In an analysis of re-finding using a query log by Tylor and Teevan [85], they found that re-finding queries show different characteristics from the initial query. In particular, they found that re-finding queries are typically shorter and rank the re-found URL higher, suggesting that people may have learned something from the initial search.

Elsweiler et al. [31] [32] [37] studied search behavior for email re-finding based on a study of 47 participants. They found several factors that affect the difficulty of the re-finding task, including the time lapsed since the message was received, the recipient and user's filing strategy. Based on the analysis of query and click patterns, they conclude that many queries are targeted for the same document, especially if the time between queries is small, and that information about people plays an important role in email queries. In terms of the search strategy, they also found that people mostly use a combination of search and browsing for re-finding. One of their conclusions is:

> Orienteering behavior was a common re-finding strategy for our participants. Most queries were very short and often consisted of partial words or names. Further, the number of hits returned by the query was not a good indicator of performance, and there tended to be a large number of message clicks per query submitted. All of this suggests that the preferred method of re-finding was to narrow the search space with a short query and browse for clues that facilitated navigating to the email required.

The design of the proposed retrieval framework supports such user behavior. We assume that users will type in a short query to narrow the search space, and in case the item is not found, then will use associative browsing to reach the target document.

2.4 Semi-structured Document Retrieval

For the term-based search model described in Chapter 3, related work can be mostly found in the investigations of the semi-structured document retrieval task, which have been tried from both IR and database perspectives. Another related area is the research on keyword search over relational databases, where the task is the ranked retrieval of structured data using keyword queries.

For semi-structured document retrieval, people have adapted traditional retrieval models to handle documents with multiple fields. Early work treated each field as a smaller document and simply combined field-level scores using linear combination or a mixture of probability models [64]. This straightforward combination of field-level scores was found to have limitations, resulting in efforts such as BM25F [70]. Recently, an adaptation of the score combination and smoothing method was suggested [91] for the language modeling approach to IR, based on the search engine Indri [63] which supports combining evidence from multiple fields.

INEX is a major initiative for the study of XML retrieval [4]. The INEX ad-hoc track addresses the task of retrieving XML data with explicit document structure, such as section and title, and has used test data consisting of scientific papers or Wikipedia articles. A recent paper from INEX [60] suggested an extension of the classic probabilistic retrieval model where each term score is weighted by tag (element type) score. A tag score for each term is estimated based on the probability that the element judged relevant contains the term.

The database community has also studied XML retrieval with keyword queries. The concept of Lowest Common Ancestor (LCA) [36] has been proposed to answer keyword queries, where the LCA corresponds to the lowest-level XML element which contains all query words in its descendant elements. Besides XML retrieval with keyword queries, there has been work about keyword search in relational databases, which includes DBXplorer [2], DISCOVER [39]. For these systems, the answers to the keyword query are the tuple trees joined from multiple tables containing query words. Another recent work adopted the relevance model for database retrieval [20].

Petkova and Croft [65] showed that a keyword query can be refined into a structured query by mapping each query term into a set of structural fragments and transforming these fragments into the XPath query that represents the original information need most appropriately. Calado et al. [13] describe a method of ranking candidate structured queries that is similar to the PRM-S retrieval model described in Section 3.2.3, although it was applied and evaluated differently.

Compared to the previous work on structured document retrieval, which focuses on the improvement in the term weighting, the proposed retrieval framework is different in that we focus on the modeling and the estimation of per-term field weights. Compared to the previous work on keyword search over XML or relational databases, we use the document as the fixed unit of retrieval and assume no hierarchical structure within each document. However, it would be an interesting direction for future research to extend the proposed retrieval model to documents with hierarchical structure.

The modeling of field relevance can be considered as an extension of many efforts to model some aspect of relevance. The relevance-based language model [53] is a well-known model of topical relevance. Here, a relevance distribution is estimated from top-k retrieved documents, which is in turn used to enrich the initial representation of the information need given as a query.

As an extension of this work, Lavrenko et al. [54, 90] introduced the structural relevance model, which estimates a term-based relevance model per field. For retrieval, they combine field-level scores based on relevance models into a document score using fixed weights. Since our work focuses on estimating per-field and per-term weights, their model can be potentially improved based on the results here. However, they focus on modeling term-level relevance, whereas our work focuses on per-term field relevance.

2.5 Federated Search

In the context of federated search and distributed IR, researchers have proposed many methods of scoring collections against a given query. Approaches such as CORI [14] and KL-Divergence [77] treat collections as large documents and apply document scoring techniques for scoring collections. Other methods, such as ReDDE [76], model the distribution of relevant documents for each collection. A recent survey can be found in [75].

Recently, Arguello et al. proposed a classification approach [5] [6] where many sources of evidences can be combined for mapping a user's query into one or more collections. Our combination approach for type prediction in Section 3.3 is similar to this work but we use features and evaluation methods more suitable for our problem domain.

In the context of personal metasearch, Thomas et al. [83] compared several server selection methods using documents collected from various sources, concluding that a selection method based on Kullback-Leibler divergence [77] performed the best. The work in Chapter 3 extends this work by proposing a type prediction method that exploits the field structure and a combination method whose performance can be improved by interaction with the user.

2.6 Associative Browsing

Since the early days of IR, researchers have been interested in the combination of search and browsing for accessing document collections. Lucarella [59] and Cimino et al. [19] described a system with a network of concepts and documents which provides search and browsing capability in a complementary manner. The I^3R system developed by Croft and Thompson [24] also assumes a scenario where documents returned by a user's initial query provide a starting point for subsequent browsing. Another paper by Croft and Turtle [25] demonstrated, in the context of document retrieval, the effectiveness of using inference networks to model the link structure between documents and using citation links instead of content-based nearest neighbor links.

Allan et al. [3] and Leuski and Allan [56] describes a system that presents the user with ranked lists and a visualization of inter-document similarities, and found that retrieval effectiveness substantially improves by doing so. Kaplan et al. [43] described a navigation scheme that adapts to user behavior. Smucker and Allan[78] found that similarity browsing can improve retrieval effectiveness when used as a search tool. Compared to these systems, our proposed approach in Chapter 4 is novel in that it suggests a feature representation of links between items. The weights of these links are trained using the click feedback from the user. We also used a simulation technique to evaluating the role of browsing in the context of known-item finding, including a model of user's knowledge, and employ different parameters of the user behavior.

Associative browsing models for personal information were introduced in previous studies [16] [15] [29]. The work in Chapter 4 improves on previously suggested models of associative browsing in that we use more general measures of associations (e.g. textual similarity and co-occurrence), while previous models defined links only between a limited set of items. From the evaluation perspective, this work is different in that we evaluated our system using a game-based user study. Our evaluation method allowed us to test our system in a controlled environment, and the data we collected can be used by other researchers without any privacy concerns.

Techniques for finding related documents or concepts have been proposed in many contexts. Danushka et al. [12] measured the semantic similarity based on the results from a web search engine. Smucker et al.[78] used the unigram language model of a given document as a query to find similar documents. The suggested model for associative browsing is novel and comprehensive, in a sense that it uses click-based training to learn the associations between information items. Using a different set of features, our learning framework can be applied to other domains.

2.7 Simulated Evaluation for Information Retrieval

Simulated evaluation has recently received much attention in many areas of IR, where it was used to replace or complement more traditional evaluation methods, such as batch experiments or user studies. Here we discuss several variants of the simulated evaluation: simulated interaction, game-based evaluation and crowdsourced evaluation.

While it has been conventional for PIR systems to be evaluated based on user studies, in the human-computer interaction literature, a highly cited paper written by Greenberg et al. [35] argues that user studies should be employed with caution. They added that the choice of evaluation methodology must arise from and be appropriate for the actual problem or research question under consideration. We believe that the same principle holds true for the evaluation of PIR. User studies and simulation techniques each have different pros and cons, and the choice or the combination of these techniques should depend on the nature of research questions at hand.

With regard to simulated evaluation for IR, Ruthven [71] evaluated the effectiveness of interactive query expansion using simulation. White et al. [88] used search simulation model to evaluate term selection methods for implicit feedback. For the evaluation of known-item search, Azzopardi et al.[7] suggested a query generation method, which is adapted in the Pseudo-desktop method described in Section 5.3. Smucker et al.[78] and Lin et al. [57] evaluated the associative browsing model using a simulated model of the user in biomedical domain.

Human computation games [86] have recently become popular as a method for obtaining a large amount of human annotations in a way that motivates participants. In the context of IR research, Ma et al. [61] introduced PageHunt, which is a game designed to collect web search log data by asking participants to find pages that they were shown. The game-based evaluation method suggested in Section 5.5 is different from the PageHunt in that we designed a game in which search and browsing are supported at the same time. We also analyzed session logs to gain insights into the use of the system, whereas previous work mostly used the data at the query level.

CHAPTER 3
FIELD-BASED SEARCH MODELS

With the popularization of web search, search has become one of the most widely used methods for personal information retrieval (PIR) as well. While search is not always possible or desirable in PIR [10], the widespread use of desktop search and the search over the user's mobile device indicate that search would continue to be an important access method for PIR.

In this chapter, we introduce field-based search models for PIR, which is composed of a retrieval framework and a set of techniques within the framework. We call them field-based search models, since we extensively use the field structures of documents to improve the effectiveness of the proposed techniques.

Although we mostly focus on the retrieval of personal information for this thesis, the application of the proposed techniques is not limited to personal information. In fact, we evaluate the field-based retrieval models proposed here using various structured document collections in Section 6.3. Also, the proposed field-based type prediction method is applicable to arbitrary sets of structured document collections.

In what follows, we first introduce our retrieval framework. We then focus on field-based retrieval methods, where we introduce two novel retrieval models – the Probabilistic Retrieval Model for Semi-structured Data (PRM-S) and the Field Relevance Model (FRM). We then shift our focus into type prediction methods, introducing a field-based and a learning-based type prediction methods.

The following notations will be used throughout this chapter. We assume that a query $Q = (q_1, ..., q_m)$ is composed of m words and each collection C contains

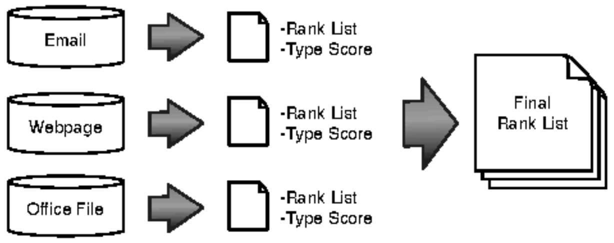


Figure 3.1. Suggested retrieval framework for desktop search.

documents with n field types $(F_1, ..., F_n)$ where n can be different for each collection. Model-specific parameters will be explained as they appear.

3.1 Retrieval Framework for Personal Information Retrieval

We now introduce a retrieval framework for PIR. A retrieval framework comprises the stages of calculation for getting the retrieval results in various forms, and it becomes the foundation of a retrieval system. In designing the framework, we take into account several characteristics of personal information.

Firstly, as personal information is composed of documents of many types, we introduce a retrieval framework where type-specific retrieval results are merged into the final ranked list. Also, since personal documents increasingly contain rich structural metadata, we use field-level evidence extensively to improve retrieval performance.

In our framework, as depicted in Figure 3.1, for each sub-collection corresponding to each document type, a ranked list is derived using an appropriate retrieval method. Then, a type prediction method calculates relevance scores for each of sub-collections. Finally, type-specific results (a set of ranked lists) and type scores are merged into a final ranked list.

Here we assume that the ranked lists are merged to present the final results in the form of a single ranked list, which facilitates the evaluation based on standard IR metrics. However, there can be other ways of presenting the results, as shown in

Figure 1.1. One alternative would be presenting the results from each document types separately. Since the type prediction scores can be used to determine the ranking among document types in that scenario as well, our framework is not confined to the presentation of a merged ranking.

While the suggested framework is composed of several stages, an alternative would be eliminating the distinction between the types, and running the retrieval over the entire collection at once. Compared to this monolithic approach where documents of all types are put into a single collection and retrieved without any consideration for the type, the proposed framework allows the use of specialized ranking features for each type. This is important for collections that have unique ranking criteria, such as a thread-based features for email [23]. Also, type prediction scores can provide extra information for ranking in addition to the document-level scores.

For the rest of this chapter, we describe the retrieval methods and the type prediction methods we propose. In order to combine the retrieval and type prediction scores, we use the well-known CORI algorithm for merging [14].

$$C'_i = (C_i - C_{min})/(C_{max} - C_{min}) \tag{3.1}$$

$$D' = (D - D_{min})/(D_{max} - D_{min}) \tag{3.2}$$

$$D'' = \frac{D' + 0.4 \cdot D' \cdot C'_i}{1.4} \tag{3.3}$$

Here, C'_i and D' are normalized collection and document score, computed using the maximum and minimum of collection scores (C_{max} / C_{min}) and document scores (D_{max} / D_{min}), respectively. Given C'_i and D', the final document score D'' can be computed by combining these two scores.

3.2 Field-based Retrieval Methods

The first step in our retrieval model is ranking documents from each sub-collection. Although any ranking criteria appropriate for each collection can be used, instead of delving into the collection-specific ranking features, we focus on developing a retrieval model which can generalize across collections.

As we reviewed in Section 2.4, it has been known that document structure can be helpful in improving retrieval effectiveness. Since each collection has structure in the form of metadata fields, here we focus on retrieval models designed for semi-structured document retrieval. We first review existing methods for structured document retrieval, and then propose two retrieval models that address the limitations of existing methods.

3.2.1 Existing Retrieval Methods for Structured Documents

Here we introduce existing retrieval methods for structured documents, including Document Query-likelihood, BM25F and the Mixture of Field Language Models. We then discuss similarities among these models, thereby deriving a general form of existing retrieval models.

3.2.1.1 Document Query-Likelihood

Document Query-Likelihood (DQL) is a standard retrieval model in the language modeling approach to information retrieval, where each document is ranked by the likelihood of generating a given query.

$$P(Q|D) = \prod_{i=1}^{m} P_{QL}(q_i|D) \tag{3.4}$$

Although the DQL method does not take the structure of documents into account, we can use DQL for our situation by ignoring the field structure and treating the whole document as a bag of words.

3.2.1.2 BM25F

Robertson et al. [70] introduced the BM25F retrieval model as a modification of the BM25 model where field-level evidence is combined at the raw frequency level. The BM25F score $BM25F(Q, D)$ is calculated as:

$$BM25F(Q,D) = \sum_{q_i \in Q} idf(q_i) \frac{Score(q_i, D)}{k_1 + Score(q_i, D)} \tag{3.5}$$

where term score $Score(q_i, D)$ is calculated as:

$$Score(q_i, D) = \sum_{F_j \in D} \frac{w_j tf(q_i, F_j, D)}{(1 - b_j) + b_j \frac{length(F_j, D)}{length(F_j, C)}} \tag{3.6}$$

Here, *idf* indicates global inverse document frequency, *tf* and *length* denotes per-field term frequency and length, respectively. Also, a field weight parameter w_j is used to combine field-level frequency into document-level frequency, and another field-level parameter b_j controls the degree of length normalization. Robertson et al. [70] suggest training w_j and b_j based on held-out queries.

3.2.1.3 Mixture of Field Language Models

Ogilvie et al. [64] suggested a mixture of field language models by linear interpolation (MFLM) for known-item search in structured document collections. A document score in the MFLM is calculated by taking the weighted average of field-level query-likelihood scores as follows:

$$P(Q|D) = \prod_{i=1}^{m} \sum_{j=1}^{n} w_j P(q_i | F_j, D) \tag{3.7}$$

MFLM also has per-field weights w_j, which are estimated based on maximizing retrieval performance in training queries, similarly to BM25F.

3.2.1.4 General Form of Existing Retrieval Models

Although these retrieval models are based on different retrieval paradigms and assumptions, they are similar in terms of how they use field weighting to combine field-level evidences. More formally, we can write a general form of existing retrieval models for structured documents as follows:

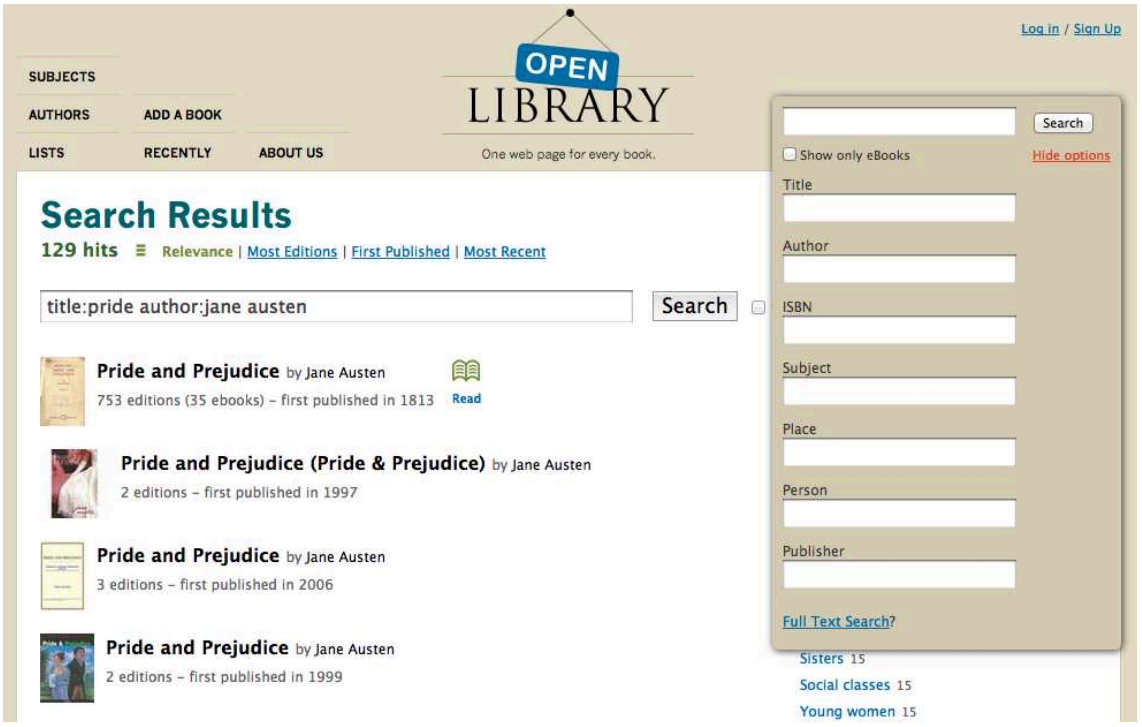
$$Score(D, q_i) = \sum_{j=1}^{n} w(F_j, q_i) Score(q_i, F_j, D) \tag{3.8}$$

$$Score(D, Q) = \prod_{i=1}^{m} Trans\Big(Score(D, q_i)\Big) \tag{3.9}$$

$Score(q_i, F_j, D)$ denotes the field-level score of D for query term q_i and field F_j. The score is a field-level term frequency in BM25F and the likelihood of observing q_i in a smoothed field-level language model of D in MFLM.

The field-level score is then combined into a document-level score weighted by $w(F_j, q_i)$, and we can write $w(F_j, q_i) = w(F_j)$ for BM25F and MFLM since their field weights are not dependent on a query term. Finally, the function $Trans$ denotes a linear transform by which weighted term frequency is transformed into a BM25 score in BM25F, and $Trans$ is an identity transform in MFLM.

Based on the generalized functional form above, we can see that all the existing models for structured document retrieval have a field weighting component, where a linear combination is used to combine field-level evidences. Previous work [70] [64] has shown the empirical effectiveness of these field weighting techniques, although they are limited in that the field weighting in these models required fixed per-field weights. Another limitation is that none of these models provided a way to incorporate relevance feedback or pseudo-relevance feedback for field weighting. That is, there has not been a natural way to adjust field weights based on the observation of relevant documents, or some approximation of them.



Search Results

129 hits ≡ Relevance | Most Editions | First Published | Most Recent

title:pride author:jane austen

Pride and Prejudice by Jane Austen
753 editions (35 ebooks) – first published in 1813

Pride and Prejudice (Pride & Prejudice) by Jane Austen
2 editions – first published in 1997

Pride and Prejudice by Jane Austen
3 editions – first published in 2006

Pride and Prejudice by Jane Austen
2 editions – first published in 1999

Figure 3.2. An example of search interface of a digital library (http://openlibrary.org) that exploits metadata fields.

In what follows, we introduce two novel retrieval methods that addresses the limitations above. The first retrieval model (PRM-S) introduces per-term and per-field weighting which are estimated using classification techniques. The second retrieval model (FRM) introduces the notion of relevance applied to field weighting, and the combination-based estimation techniques for field relevance.

3.2.2 Evidence for Per-term Field Weighting

Before we describe the proposed retrieval models, we provide justifications for per-term field weighting. Figure 3.2 shows a search interface which allows various types of field-level interactions. Here, users can type in field operators to specify the field they associate with each query term. Alternatively, users can use an advanced search interface to specify the same kind of information.

Table 3.1. Distribution of Query Fields in Email Known-item Search (from Elsweiler et al. [32])

Field	Count	Percentage
subject	231	7.49
sender	585	18.97
subject or send (default)	1677	54.38
entire message	412	13.36
to or cc	179	5.8

A recent study of email search behavior [32] investigated the distribution of fields for query terms. The results, shown in Table 3.1 indicates that people's search terms are associated with different fields. Besides the *subject* or *sender*, which were the default fields in the search interface, the users' specification of fields was spread across different fields.

Our study of retrieval in personal social media collections [55] shows similar trends. From known-item queries for personal Twitter and Facebook collections, we obtained the set of query words manually specified with fields, and we looked at how the field information is distributed. From Figure 3.3 [1], it can be observed that a majority of query words are associated with the *msg/text* (46% in Facebook and 65% in Twitter) or *uname* (32% in Facebook and 11% Twitter) fields. Words specified with *cmtmsg* (15%) or *re-text* (15%) also share a fair portion of all the fields. Overall, query terms were associated with various fields.

These data suggest that users tend to associate various fields with query terms. The problem here is that most users still issue plain keyword queries because typing structured queries is not only cumbersome, but in many cases impossible because it requires the knowledge of the underlying data schema. In what follows, we propose retrieval models that eliminate the need for expressing structural intent by predicting the user's query intent and incorporating it into the retrieval model. In effect, the

[1]Fields that are never specified are omitted in the figures.

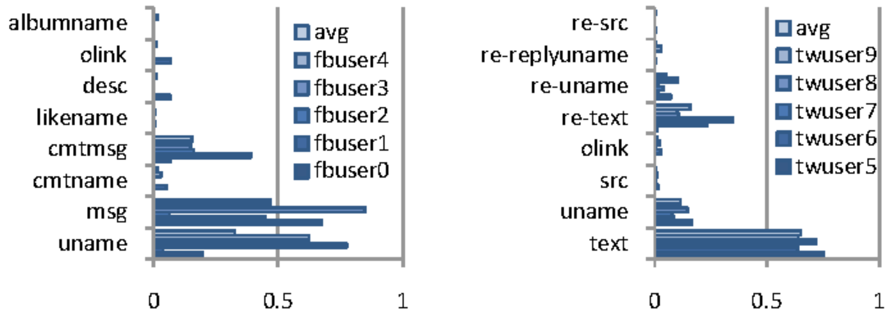


Figure 3.3. Field distribution for manual-specified queries for Facebook (left) and Twitter (right).

proposed retrieval models aims to achieve the same results, while removing the needs to specify the fields.

3.2.3 Probabilistic Retrieval Model for Semi-structured Data

The probabilistic retrieval model for semistructured data (PRM-S) [51] scores documents by combining field-level query-likelihood scores similarly to other field-based retrieval models [64]. The main feature of the PRM-S model is that weights for combining field-level scores are estimated using the predicted mapping between query terms and document fields, which can be efficiently computed from collection term statistics.

More formally, using Bayes' theorem, we can estimate the posterior probability $P_M(F_j|w)$ that a given query term w is mapped into document field F_j by combining the prior probability $P_M(F_j)$ and the probability of a term occurring in a given field type $P_M(w|F_j)$.

$$P_M(F_j|q_i, C) = \frac{P(q_i|F_j, C)}{\sum_{F_k \in F} P(q_i|F_k, C)} \tag{3.10}$$

Here, $P_M(w|F_j)$ is calculated by dividing the number of occurrences for term w by total term counts in the field F_j across the whole collection. Also, $P_M(F_j)$ denotes the prior probability of field F_j mapped into any query term before observing collection statistics.

With the mapping probabilities estimated as described above, the probabilistic retrieval model for semistructured data (PRM-S) can use these as weights for combining the scores from each field $P(q_i|F_j, D)$ into a document score, as follows:

$$P(Q|D) = \prod_{i=1}^{m} \sum_{j=1}^{n} P_M(F_j|q_i) P(q_i|F_j, D) \tag{3.11}$$

This model was shown to have better performance than other field-based retrieval methods, such as the mixture of field language models [64] and BM25F [70], for a semi-structured document retrieval task using the IMDB [51] and TREC email [46] collections. We present these experimental results in Section 6.3.

3.2.3.1 Mixture of PRM-S and Document Query Likelihood

An important assumption of the PRM-S is that each query term is chosen from a specific document field. However, it may not make sense to assume that users choose every query term with a particular field in mind. In this aspect, the PRM-S may seem too extreme since it only considers field-level scores and totally disregards document scores. A simple yet effective solution for this problem is to interpolate PRM-S with the document query likelihood model (PRM-D) as in Equation 3.12, thereby striking a balance between these two.

$$P(Q|D) = \prod_{i=1}^{m} ((1 - \lambda) \sum_{j=1}^{n} P_M(F_j|q_i) P(q_i|F_j, D) + \lambda P(q_i|D)) \tag{3.12}$$

where λ is the parameter that controls the interpolation ratio.

3.2.4 Field Relevance Model

Previous retrieval models discussed so far used several sources to estimate field weights. However, there has not been a natural way to incorporate relevance feedback for the estimation. In other words, even when a set of relevant documents were known, it was not easy to exploit this information for retrieval. Also, while the PRM-S model employed per-term field weights, the estimation is based on limited sources.

To address the limitations of existing retrieval models as described above, here we introduce the notion of field relevance and corresponding retrieval model (the field relevance model). We investigate how field relevance can be estimated either when relevant documents are known (relevance feedback) or not (pseudo-relevance feedback). We then prove that the field relevance model with relevance feedback gives an optimal set of field weights.

3.2.4.1 Field Weighting as Field Relevance

The notion of relevance is central to the area of information retrieval, yet the multi-faceted nature of relevance led to many definitions and controversies. Although there has been numerous efforts [52, 53, 69] to model and incorporate the relevance in a retrieval model, most have focused on modeling relevance in bag-of-word retrieval models without using the document structure.

In structured document retrieval, however, the fields within each document encode different aspects of information, and we can also find implicit structure within a user's keyword query. Given the structure found in both queries and documents, we can argue that the degree of topical relevance depends on the matching of the structure as well as terms.

As an illustration, consider a query 'james meeting 2011' issued for an email collection. Assume that a user formulated this query based on the memory of an email whose *sender* is 'james', whose *subject* and *body* fields include 'meeting', and that has the term '2011' in the*date* field. A query term may have matches in the fields that user did not intend, (e.g., 'james' can be found in *body* field), but the term scores from such fields should be considered less important since those do not match with the user's structural intent.

Since traditional models of relevance feedback focused on adjusting query-term weights, they cannot capture this variations in relevance with respect to the matching between structural components of a document and a query. To overcome this limitation, it is necessary that structural components of a collection be considered in modeling relevance. We now formally define field relevance and the corresponding retrieval model in the context of a keyword query.

Field Relevance Given a query $Q = (q_1, ..., q_m)$, *field relevance* $P(F_j|q_i, R)$ is the *distribution* of *per-term* relevance over document fields.

Field Relevance Model Based on field relevance estimates $P(F_j|q_i, R)$, the *field relevance model* combines field-level scores $P(q_i|F_j, D)$ for each document using field relevance as weights.

$$Score(D, Q) = \prod_{i=1}^{m} \sum_{j=1}^{n} \hat{P}(F_j|q_i, R) P(q_i|F_j, D) \qquad (3.13)$$

From the users' perspective, field relevance can be regarded as their per-term query intent over document fields. Alternatively, we can interpret field relevance as the generalization of field weighting components that are found in existing retrieval models. It is dependent on both word and document fields, unlike the per-field weights of BM25F and MFLM.

The field relevance as defined above looks similar to the mapping probability $P_M(F_j|q_i)$ in PRM-S. However, while the estimation of the mapping probability (per-term field weights) in PRM-S is conceptually based on a classification framework, we interpret field weights as a new aspect of relevance, and we argue that this opens up new possibilities for estimation. In Section 3.2.4.3, we also show how the mapping probability can be incorporated to improve the estimation of field relevance.

3.2.4.2 Field Relevance Estimation by Relevance Feedback

Based on our definition of field relevance, we discuss how relevance feedback can be incorporated into the existing structured document retrieval framework. Ideally, if we assume knowledge of relevant documents, we can directly use the language model of relevant documents to estimate field weights.

$$P(q_i|F_j, R) := P(q_i|F_j, D_R) \tag{3.14}$$

In other words, the term distribution of known relevant documents across different fields indicates the relevance of each field for a given query term. Going back to our earlier example on the query 'james meeting 2011', if we knew in which fields in the relevant email the query terms are located, we could easily identify relevant fields for each query term. As this is based on the observation of relevant documents, we call this the 'oracle' field weight estimate in what follows.

Since we use the field relevance as field weights in our retrieval model, this allows *true* relevance feedback in field weighting — knowledge of relevant documents can be naturally incorporated into the estimation. This suggests the possibility to improve retrieval effectiveness if a user is willing to provide relevance judgments.

In more practical scenarios, where relevance judgments are not available, we need to find alternative sources by which we can approximate the field-level term distribution of relevant documents. One way is to use the top-k retrieved documents as the approximation of relevant documents, as was done in previous work [53]. In this thesis, to improve the quality of estimation further, we combine this with other sources of estimation, as will be discussed in what follows.

3.2.4.3 Field Relevance Estimation by Combining Sources

In the previous section, we introduced the notion of field relevance as the generalization of field weights, and described how we can estimate field relevance when relevant documents are known. In practice, field relevance needs to be estimated based on the information available without the knowledge of relevant documents. Given the size of the parameter space, however, it is challenging to estimate the value per field and query term.

To address this concern, we introduce a learning framework where field relevance can be estimated based on the combination of several sources. Since each source gives the distribution of field relevance for each query term, we have only as many parameters as the number of sources.

Here we introduce our estimation framework more formally. We first define the field relevance estimate $\hat{P}(F_j|q_i, R)$ as a linear combination of several sources. Here, $\Lambda = (\lambda_1, ...\lambda_p)$ denotes weights used for the mixture.

$$\hat{P}(F_j|q_i, R) = \sum_{k=1}^{p} \lambda_k P_k(F_j|q_i) \tag{3.15}$$

We now present our framework for estimating field relevance based on the combination of several sources. First, we need to find a reasonable set of weights $\Lambda = (\lambda_1, ...\lambda_p)$ to combine sources into a final estimate of field relevance. If we assume that we have training queries with relevance judgments, we can use coordinate ascent search [62, 9] to find a set of parameters that maximize the target metric in the training queries. Since we have only 5 parameters, this is computationally tractable. As for the choice of target metric, we followed previous work [62, 9, 70, 64], which used metrics of retrieval effectiveness, such as MAP or NDCG.

In what follows, we introduce the sources we employed to estimate field relevance. As we employ some of field weight estimates from previous work as sources and the combined estimates are used as field weights within the field relevance model, it is fair to say that the field relevance model is a generalization of existing retrieval models for structured documents.

3.2.4.3.1 Collection Language Model As introduced in Section 3.2.3, PRM-S estimates per-field and per-term weights based on collection statistics. This is a reasonable choice assuming that the field-level term distribution of relevant documents will be similar to that of the collection. It also explains the empirical effectiveness of PRM-S [51, 46].

In our framework, we incorporated as a source the likelihood of observing a query term q_i in the unigram field language model of the collection.

$$P(F_j|q_i, C) = \frac{P(q_i|F_j, C)}{\sum_{F_k \in F} P(q_i|F_k, C)} \tag{3.16}$$

While this unigram language model was shown to be effective in previous evaluation with PRM-S [51, 46], it is limited in that it ignores the dependencies between query terms. To address this problem, we use a field-level bigram language model whose probability is dependent on the previous query term as well as the current query term.

$$P(F_j|q_i, q_{i-1}, C) = \frac{P(q_i, q_{i-1}|F_j, C)}{\sum_{F_k \in F} P(q_i, q_{i-1}|F_k, C)} \tag{3.17}$$

3.2.4.3.2 Top-k Retrieved Documents To approximate the field-level term distribution of relevant documents, we described how a field-level collection language model can be used as a source of estimation. However, in many cases the field-level term distribution of relevant documents will diverge significantly from that of the collection. We somehow need ways to approximate the term distribution of relevant documents more closely.

To solve this problem, we propose using the top-k retrieved documents for a given query. Specifically, we combine the field-level language models of documents retrieved by some ranking methods to build a new language model for each field, and use it to approximate per-field and per-term weights:

$$P(F_j|q_i, D_{TopK}) = \frac{P(q_i|F_j, D_{TopK})}{\sum_{F_k \in F} P(q_i|F_k, D_{TopK})} \tag{3.18}$$

The idea of using the top-k retrieved documents to approximate some aspect of relevance was introduced in Lavrenko and Croft [53], and our approach is similar in that we use top-k retrieved documents to approximate some dimension of relevance. The difference is that we use it to estimate field relevance, whereas their goal was to estimate term weights.

We use similar techniques to build the field language model of top-k documents as in previous work [53]. The probability is estimated based on the weighted average of the top-k retrieved documents, where the weights are the query-likelihood scores for those documents:

$$P(q_i|F_j, D_{TopK}) = \sum_{D \in TopK} P(w|F_j, D) \prod_{i=1}^{n} P(q_i|D) \tag{3.19}$$

Similarly to the case of a collection language model, we use bigram language models of the top-k documents to estimate field relevance.

$$P(F_j|q_i, q_{i-1}, D_{TopK}) = \frac{P(q_i, q_{i-1}|F_j, D_{TopK})}{\sum_{F_k \in F} P(q_i, q_{i-1}|F_k, D_{TopK})} \tag{3.20}$$

3.2.4.3.3 Per-Field Weights based on Training Queries Previous field-based retrieval models [70, 64] introduced ways of estimating per-field weights based on the retrieval effectiveness in training queries. Although field relevance in this work is defined to be dependent on each query term as well as field, we can incorporate these per-field weights as one of the sources to increase the reliability of the estimation.

3.2.4.4 A Mathematical Justification

In order to justify the formulaion of field relevance model, we provide another perspective of the field relevance model introduced in the previous section, based on a vector space interpretation of field weighting. We first explain how field weighting can be considered as a vector projection, and show that the weight vector and the score vector should have the same direction in order to maximize the score of a given document.

We finally prove that the field weighting based on the oracle field relevance estimate (true relevance feedback) results in the condition under which the score of a given relevant document is greater than in any choice of field weight vector.

3.2.4.4.1 Field Weighting as a Projection The retrieval model we described in the previous section, as well as all the other existing field-based retrieval models, employs a linear model for field-level score combination. Now we show that field weighting in general can be considered as a projection under this condition.

Let's assume a document D_k with n fields that has a field-level score vector $\vec{s_k}$ of n dimensions for a given query term $q_i \in Q$. Further assume that a field weight vector $\vec{w}$ of n dimensions is used to combine field-level scores into a document-level score. Given this notation, we can represent the combination as a dot product as follows:

$$Score(D_k, q_i) = \vec{s_k} \cdot \vec{w} \tag{3.21}$$

If we use θ to denote the angle between $\vec{s_k}$ and $\vec{w}$, we can re-write the dot product as follows:

$$Score(D_k, q_i) = |\vec{s_k}||\vec{w}|cos\theta \tag{3.22}$$

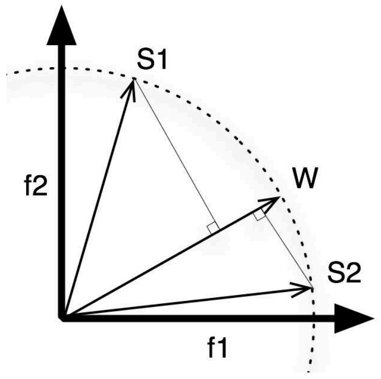


Figure 3.4. A vector space interpretation of field weighting with $\vec{w}$ for two documents D_1 and D_2 with field-level score vector $\vec{s_1}$ and $\vec{s_2}$, respectively.

Based on the simple derivation above, we can see that the $Score(D_k, q_i)$ depends on both the magnitude of the score vector $|\vec{s_k}|$ and the $cos\theta$. For comparing $Score(D_k, q_i)$ across different documents, we can ignore $|\vec{w}|$ as it is not dependent on documents.

This leads to the following formula, which shows that the document-level score for query term q_i is rank-equivalent to the magnitude of vector projection of the score vector $\vec{s_k}$ onto the weight vector $\vec{w}$:

$$Score(D_k, q_i) \overset{rank}{=} |\vec{s_k}| cos\theta \tag{3.23}$$

In other words, we can regard the weighted combination of field-level scores as the projection of the score vector onto the weight vector, and this shows how field weighting impacts document scoring within linear combination.

To illustrate this point, we provide an example with two documents D_1 and D_2 with two fields in Figure 3.4, where we represented the weight vector $\vec{w}$ and the score vector $\vec{s_1}$ and $\vec{s_2}$ of two documents in a unit circle. In this example, we can see that the direction of the weight vector $\vec{w}$ is set closer to the score vector of document D_2, and the resulting score for D_2 is greater than that of D_1, because $\vec{s_2}$ has a smaller angle to $\vec{w}$ (larger $cos\theta$) compared to $\vec{s_1}$.

3.2.4.4.2 Maximizing Score of a Document We now consider an *optimal* weight vector — the one that maximizes the score of a relevant document. Let's assume that we have observed a relevant document D_R with a score vector $\vec{s_R}$ for a query term $q_i \in Q$.

Our goal is to derive a weight vector $\vec{w_o}$ that will rank D_R as highly as possible. Following the view of field weighting as a projection, we can see that the relative score of the document D_R is maximized when the angle θ_R between $\vec{s_R}$ and $\vec{w}$ is 0, which gives $cos(\theta_R) = 1$:

$$max(\vec{s_R} \cdot \vec{w}) = |\vec{s_R}||\vec{w}|cos(\theta_R) = |\vec{s_R}||\vec{w}| \tag{3.24}$$

This can be achieved when $\vec{w}$ has the same direction as the score vector $\vec{s_R}$ of D_R, as seen below:

$$\arg\max_{\vec{w}} Score(D_R, q_i) = \frac{\vec{s_R}}{|\vec{s_R}|} \tag{3.25}$$

In other words, this choice of weight vector is *optimal*, since no other choice of weight vector $\vec{w}$ will give higher relative score for D_R than $\vec{w_o}$.

There are several considerations in the notion of *optimality* above. Firstly, $\vec{w_o}$ may not always give the ranking where the relevant document would be ranked at the top position. Rather, it scores the relevant document as highly as possible for the query term q_i.Another point is that we only considered field weighting per query term above. Such weighting is not guaranteed to give the best performance when per-term scores are combined into a final score.

Finally, note that our discussion so far is based on the context of having a single relevant document, which includes scenarios such as homepage or known-item finding. However, it can be easily extended to accommodate the cases with multiple relevant documents. If we can represent each relevant document as a vector of per-field scores, we can take the vector sum of these to get $\vec{s_R}$, from which the *optimal* field weights can be derived.

3.2.4.4.3 Relevance Feedback in Field Relevance Model Based on the discussion above, we now consider the case of true relevance feedback in the field relevance model. Specifically, we show that the field relevance model shown in Section 3.2.4 gives the highest per-term score for a given relevant document D_R:

$$P(Q|D) = \prod_{i=1}^{m} \sum_{j=1}^{n} P(F_j|q_i, D_R) P(q_i|F_j, D) \tag{3.26}$$

Consider the following derivation of the retrieval model in Equation 3.26. We use a Bayesian transformation of $P(F_j|q_i, D_R)$, then eliminated the prior $P(F_j|D_R)$ by assuming that it is uniform:

$$\begin{aligned} P(Q|D) &\overset{rank}{=} \prod_{i=1}^{m} \sum_{j=1}^{n} P(q_i|F_j, D_R) P(F_j|D_R) P(q_i|F_j, D) \\ &\overset{rank}{=} \prod_{i=1}^{m} \sum_{j=1}^{n} P(q_i|F_j, D_R) P(q_i|F_j, D) \end{aligned} \tag{3.27}$$

Now, if we are to score D_R using the retrieval model above, we can see that the field scores and weights take the same value (that is, field-level query-likelihood) for this particular document D_R:

$$P(Q|D_R) = \prod_{i=1}^{m} \sum_{j=1}^{n} P(q_i|F_j, D_R) P(q_i|F_j, D_R) \tag{3.28}$$

The vector space interpretation of field weighting in Section 3.2.4.4.2 proves that this is the weight vector $\vec{w}(F, q_i)$ that gives higher rank to D_R than any other choice of $\vec{w}$. Therefore, we can argue that the proposed retrieval model is sensible in that it gives the highest advantage to the relevant document given the knowledge of the document.

3.2.4.5 Similarity Metrics for Field Relevance

Finally, we introduce several measures of similarity between two estimates of field relevance. We define these metrics in terms of a per-term similarity between a given field relevance estimate and an oracle estimate, since we mostly use these metrics to evaluate given field relevance estimates against oracle estimates. Per-term similarities are then averaged to become a query-level similarity.

First, since we defined field relevance as a probability distribution, we can use the **Kullback-Leibler divergence** between two per-term estimates of field relevance:

$$D_{KL}(P, P_O) = \sum_{j=1}^{n} P_O(F_j|q_i) log_2 \frac{P_O(F_j|q_i)}{P(F_j|q_i)} \tag{3.29}$$

We can also the **cosine similarity** of an oracle estimate $\vec{w_o}$ and the given estimate of field relevance $\vec{w}$. This measure is motivated from the derivation in Section 3.2.4.4.1:

$$Cos(\vec{w_o}, \vec{w}) = \frac{\vec{w_o} \cdot \vec{w}}{|\vec{w_o}||\vec{w}|} \tag{3.30}$$

Finally, if we regard the problem of field relevance estimation as the relevance ranking of fields for a given query term, we can define a **precision** measure for each query term. The value of this per-term precision measure is 1 if the field with the highest field relevance estimate matches with that of an oracle estimate, and 0 otherwise.

3.3 Field-based Type Prediction Methods

In this section, we introduce our type prediction methods in detail. The type prediction problem bears some similarity to the vertical or resource selection problem in aggregated or federated search (refer to Section 2.5 for a detailed review) in that the system tries to score the results from each vertical, resource, or collection based on predicted relevance for a given query. In this sense, all these problems can be put in a broad category of collection scoring. There are, however, several notable differences.

First, type-specific sub-collections in the desktop are *cooperative* in that all the documents are available to a single system. This means that sampling techniques used for federated search may not be necessary for desktop search; second, unlike typical collections used for aggregated search, the sub-collections in the desktop environment are small and have considerable topical overlap. This makes it challenging to apply content-based collection scoring techniques (e.g., CORI [14]) directly; third, each sub-collection in the desktop has unique metadata that has not been exploited in existing collection scoring methods.

We first describe existing methods for type prediction which are adopted from recent works on aggregated and federated search [5] [6]. Then we introduce a new type prediction method that exploits document metadata. Lastly, we explain how multiple type prediction methods can be combined using several learning methods.

3.3.1 Existing Methods for Type Prediction

Here we introduce existing type prediction methods, which are mostly adapted from collection selection techniques described in Section 2.5. These methods are used as baselines for our evaluation, and they are used as a feature for the adaptive type prediction method we describe in Section 51.

3.3.1.1 Query-likelihood of Collection

Many traditional resource selection methods (e.g. CORI) are computed from collection term statistics. Among these, we use collection query-likelihood (CQL)[77], which is a resource selection method based on the language modeling approach. The approach here is to collapse all documents in each collection into one giant 'document' and use the query-likelihood score for the document as the collection score:

$$CQL(Q,C) = \prod_{q \in Q} (\lambda P(q|C) + (1-\lambda)P(q|G)) \tag{3.31}$$

C is the language model of each sub-collection and G is the language model of the whole collection. The smoothing parameter λ adjusts the interpolation ratio of $P(q|C)$ and $P(q|G)$. CQL was shown to be the most effective among resource selection methods in a recent evaluation [83].

3.3.1.2 Query-likelihood of Query Log

Another source of evidence for the type prediction is the aggregated query terms used for finding documents that belong to each sub-collection. As done in previous work [5] [6], we use the query-likelihood score of the language model (QQL) built by queries targeted for sub-collection C as shown below:

$$QQL(Q,C) = \prod_{q \in Q} (\lambda P(q|L_C) + (1-\lambda)P(q|L_G)) \tag{3.32}$$

L_C is the language model of the query log corresponding to collection C. L_G is similarly defined using the query log across all collections.

3.3.1.3 Geometric Average

Another class of resource selection methods combine the score of top documents to evaluate each collection given the user query. Seo et al. [73] proposed using the geometric mean of the top m documents as the combination method,

$$GAVG(Q,C) = (\prod_{d \in D_{top}} P(Q|d))^{\frac{1}{m}} \tag{3.33}$$

where D_{top} is the set of top m documents from the collection and the score $P(Q|d)$ is padded with $P_{min}(Q|d)$ if fewer than m documents are retrieved.

3.3.1.4 ReDDE

ReDDE [76] [6] scores a target collection based on the expected number of documents relevant to the query. Although previous work used a centralized sample index to derive this expectation, we can estimate this directly from the target collection,

$$ReDDE(Q, C) = \sum_{d \in D_{top}} P(Q|d) \tag{3.34}$$

which is equivalent to using the sum of the top document scores belonging to each collection. Intuitively, this results in a higher score for the collection with more documents in higher positions.

3.3.1.5 Query Clarity

So far, most of our methods have been derived from resource selection techniques developed in the context of distributed IR. Query performance prediction methods can also be used for type prediction by assigning a higher score for the collection with higher predicted performance. Among such methods, we employ Query Clarity [26], which predicts performance using the KL divergence between a query language model and a collection language model.

$$Clarity(Q, C) = \sum_{w \in V} P(w|L_Q) log_2 \frac{P(w|L_Q)}{P(w|C)} \tag{3.35}$$

Here, query language model L_Q is estimated from the top m documents from the collection.

3.3.1.6 Dictionary-based Matching

In some cases, users provide direct clues about which file type they intended to search, by including terms such as ‘sender’(for email), ‘pdf’(for office document) or ‘www’(for webpage). Although these terms may not occur in a majority of queries, they can be a strong indication of type membership for a given query. We built a dictionary for each sub-collection by using the names of the collection and metadata fields.

3.3.2 Field-based Collection Query Likelihood

Although some of existing type prediction methods use the collection term statistics, none use the field structure of documents available for personal information collection. Considering that the retrieval effectiveness of semi-structured document collections has been improved by exploiting this structure [51], we can expect similar benefits for the type prediction problem.

Field-based collection query likelihood (FQL) – our new method for type prediction – extends the collection query likelihood model for collection scoring [77] by combining the query-likelihood score for each field of the collection instead of using the score for the whole collection. In other words, if we borrow the view of query-term and field mapping described in Section 3.2.3, we try to infer the mapping between a user query and each collection by combining mapping probabilities for the fields of each collection.

More formally, for a collection C that contains documents of n field types $(F_1, ..., F_n)$, we can combine the language model score of each field as follows:

$$FQL(Q, C) = \prod_{q \in Q} comb_{F_i \in C}(P(q|F_i)) \tag{3.36}$$

Here, F_i is a smoothed language model of the ith field of the collection and *comb* can be any function that can combine n numbers into one. We experimented with many variations of *comb* function and found that arithmetic mean gives the best performance.

3.3.3 Combining Type Prediction Methods

Considering that the type prediction methods introduced so far are derived from different sources, it is plausible that we can get further performance benefits by combining individual methods in a linear model where weights are found using learning methods. In this section, we describe three learning methods with different objective functions: grid search of parameter values, a multi-class classifier and a rank-learning method.

3.3.3.1 Iterative Grid-search

Since we have only seven features to be combined, It is feasible to perform a grid search of parameter values that maximize the performance of a training set of queries. Specifically, we can find the optimal value for each parameter in turns while fixing the values of the parameters previously found, and repeating the whole procedure until we reach convergence. In searching for the optimum value of each parameter, we employed Golden Section Search [67].

3.3.3.2 Multi-class Classification

Given that we want to predict one of k document types for a given query, this is typical multi-class classification scenario where each type corresponds to a class. Among many choices of such methods, we used a one-vs-rest (OVR) support vector machine classifier (MultiSVM) available in the Liblinear Toolkit[2]. Since the output of this classifier is not suitable to be used directly as type scores, we used a simple linear transform to convert the scores into probabilities.

3.3.3.3 Rank-learning Method

Alternatively, one can cast the type prediction task as a ranking problem where we try to rank the relevant collections higher than non-relevant collections. This approach can be especially beneficial for the case where the user is finding multiple documents with different types, since such a situation is hard to model with typi-

[2]http://www.csie.ntu.edu.tw/ cjlin/liblinear/

cal multi-class classification methods. RankSVM [40] was used as the rank-learning method.

3.4 Summary

In this chapter, we proposed a field-based search model for personal information retrieval where type-specific retrieval results are merged into a final rank list based on type prediction scores. As an example of the type-specific retrieval method, we introduced novel retrieval methods for structured documents called Probabilistic Retrieval Model for Semistructured data (PRM-S) and Field Relevance Model (FRM). The field relevance model introduces new estimation techniques for per-term field weighting, using either relevant documents or the combination of several sources. For the type-prediction method, we introduced a method called Field-based collection Query Likelihood (FQL), and a discriminative learning framework that combines existing type prediction methods as features.

CHAPTER 4
ASSOCIATIVE BROWSING MODEL

Modern computer systems allow us to access personal information in many ways. Apart from hierarchical file organization, keyword search has become a standard feature for many platforms. Although search can greatly ease the task of finding personal information, there are still cases in which the user's initial search attempt fails. Users may not recall search terms at all, or the initial search keywords may not be sufficient to retrieve the target document. To address these limitations, we propose associative browsing as an alternative retrieval method for personal information, and introduce a technique for enabling associative browsing of personal information.

Associative browsing, in our definition, denotes the process of browsing through personal information based on the associations between information items (e.g., documents). Each item acts as a query to retrieve related items, and this chain of associations can be followed until the information desired is found. In the literature, this mode of information access by following a series of small steps is called *orienteering*, in contrast to *teleporting* where the user directly taken to the destination (e.g., using keyword search).

Previous work show that browsing has several benefits for personal information retrieval. Studies in cognitive psychology [28] [84] show that people remember facts primarily by associations, which explains the intuitive appeal of associative browsing. Earlier works on PIM has found [68] [10] that people are particularly likely to orienteer when working with their personal information. More recently, Teevan et al. [80]

suggest that many people tend to find information by orinteering (e.g., navigating through folders) instead of teleporting (e.g., using keyword search).

However, associative browsing requires the associations between items through which users can browse their information. Unlike the web where hyperlinks provide a ubiquitous means of linking, there is no such mechanism for personal information. Therefore, in order to enable browsing support for personal information, a central issue is to create associations between items.

Researchers have tried to build such additional structure on top of personal information [44] [29] that enables associative browsing. Such techniques are generally referred to as a 'semantic desktop'—the organization of personal information across devices and applications based on concepts and the relationships between them. With a semantic desktop, instead of navigating through files and folders, users can browse their personal information based on people's names, events and the relationships between them.

Despite this appealing vision, none of the approaches suggested previously have been widely adopted. According to a recent user study [72], the most conspicuous problem is that these systems have a complex data model and interface, making them hard to understand and maintain for the end user. A related issue is that users need to make manual annotations (e.g. Tom is-a-friend-of Mary) to populate the data model. Recently, Sauermann et al.[72] found that users of such systems in general are not willing to make annotations to their database except for simple tagging.

In an effort to keep the benefit of structured data while minimizing the cost for the user, we propose a simple model of associative browsing for personal information. It is composed of (information) *items*, *tags* and the *links* between items. Items here represent information objects with textual contents. These objects can be *documents* collected from many sources, or *concepts* – entities and terms of interest to the user.

Tags and links are the metadata that enables the grouping or the association of individual items.

This model eliminates many of the complications that existing approaches suffer from. First, the fundamental unit of managing information is documents and concepts, which is a more intuitive representation of personal information than the RDF-based ontologies employed in previous work. Secondly, although the system internally maintains many types of links between items, users are presented with a single ranked list of related items for browsing.

This presentation in the form of a ranked list also makes it possible to automatically maintain the associations. By presenting a ranked list of suggestions, we can collect the user's click feedback, which is then used to refine suggestions for browsing. This allows the suggestions to be personalized according to user's click patterns. For the rest of this chapter, we first introduce an associative browsing model for personal information. We then describe a learning framework for creating associations between items.

4.1 Associative Browsing Model

In this section, we introduce the details of the assocative browsing model. We first introduce our data model, followed by a usage scenario in known-item finding. Finally, we describe a system that implements this model.

4.1.1 Data Model

At a high level, our associative browsing model is composed of information items and the associations between them. Figure 4.1 shows an UML diagram of our data model. An *item* is a fundamental unit of our data model, which can be either the *documents* collected from many sources (e.g., desktop files, emails, calendar items), or the *concepts* (e.g., person names, events, etc.). Items can be tagged, and have a text

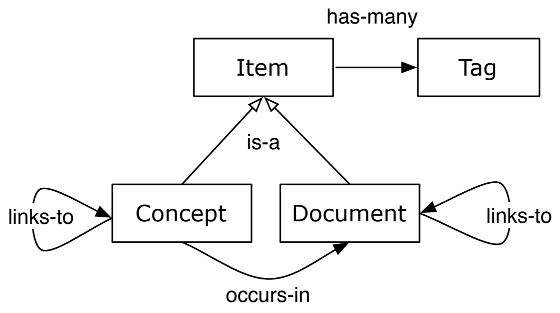


Figure 4.1. An UML diagram for for suggested associative browsing model.

representation—title, URI, content and metadata—the user can perform a keyword search for any of them. Although we proposed this model in the domain of personal information, it is a general model of associative browsing which can be used in many other domains.

One distinctive part of our model is the *concept*, which denote entities and terms of interest to the user. Concepts have an association structure of their own, and are linked to documents they are extracted from. Depending on how the model is implemented, concepts can be extracted from document metadata, or the system can allow the user to create concepts. While concepts provide another access mechanism for documents, they are optional in that using only the associations between documents is possible. In our evaluation, presented in Section 6.4, we evaluate both scenarios.

Another feature of our data model is a rich link structure between items as seen in Figure 4.2. The *associations* between concepts and documents are created based on the occurrence of a concept within a document (e.g. an *email* and its *sender*). While this provides natural connections between documents and concepts, creating associations between documents and between concepts is harder, since there is no single method that gives both high coverage and precision.

Previous research solves this problem by asking the users to create associations manually, or by extracting the associations automatically between a limited set of items. In our work, we address this issue by ranking candidates for browsing based on the combination of many similarity metrics. The top k items are then presented to the user as suggestions for browsing, and the system learns to improve suggestions based on the click feedback from the user. Figure 4.3 shows an user interface for browsing, and the framework for ranking browsing suggestions are provided in Section 4.2.

4.1.2 Applications for Known-item Finding

There can be many use cases for this associative browsing model. For instance, such a rich network of association would be suitable for exploratory search [87] in personal information. Another possible use case is known-item finding, where associative browsing can provide a back-up strategy for keyword search.

In this thesis, we focus on evaluating the associative browsing model in the known-item finding scenario, since it is the most common task in personal information access and the well-defined structure allows us to use the evaluation methods introduced in Section 5. Figure 4.2 provides an example of how associative browsing can be combined with keyword search for known-item finding. Imagine a user who is trying to find a webpage she has seen. Further assume that she cannot come up with a good keyword for search, yet she remembers the sender of a related email.

Based on proposed retrieval framework, the user can first use keyword search to find a relevant concept (person), and then browse into the target document (webpage) through another document (email) associated with both the concept and the target document. Here, dotted lines represent the associations between documents and concepts. Directed lines denote how a user can access the target webpage by using keyword search and associative browsing. Now we describe a system that implements this associative browsing model. We employ this system for the evaluation described in Section 6.4.

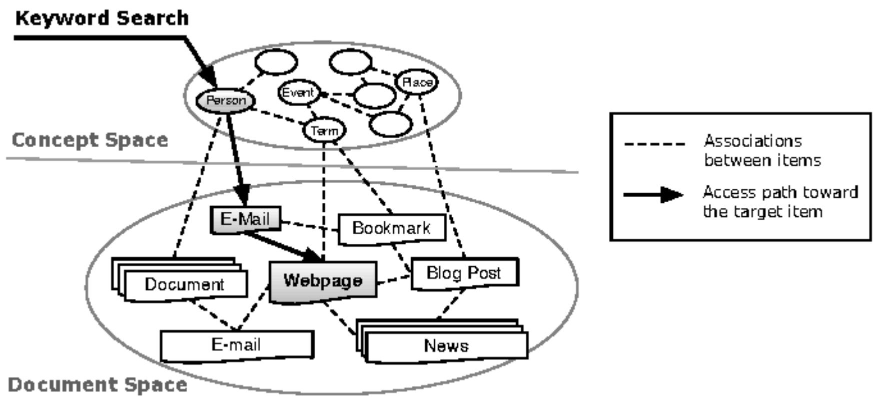


Figure 4.2. An illustration of the suggested associative browsing model. Dotted lines represent the association between documents and concepts. Directed lines denote how a user can access the target webpage by using keyword search and associative browsing.

4.1.3 System Architecture

In this section, we introduce a prototype system we developed for this project called LiFiDeA[1]. First, it collects documents and extracts concepts from document metadata. After this initialization step, the user can browse the space of concepts and documents. During the browsing, the clicks on ranked lists are logged by the system, and used for refining the suggestions.

4.1.3.1 Document Collection

First, LiFiDeA collects personal documents from various sources including desktop files, email and websites that provide RSS feed. This functionality allows the system to work with all of the users' personal information on computing clouds as well as desktop because most web services offer an RSS feed. During this collection step, the system also extracts metadata (e.g., authors and recipients of emails, tags of blog postings) associated with each document type. Several concepts are created from the metadata (e.g., senders and receivers of emails).

[1]The source code for the LiFiDeA is publicly available at http://github.com/jykim/lifidea

4.1.3.2 Concept Creation

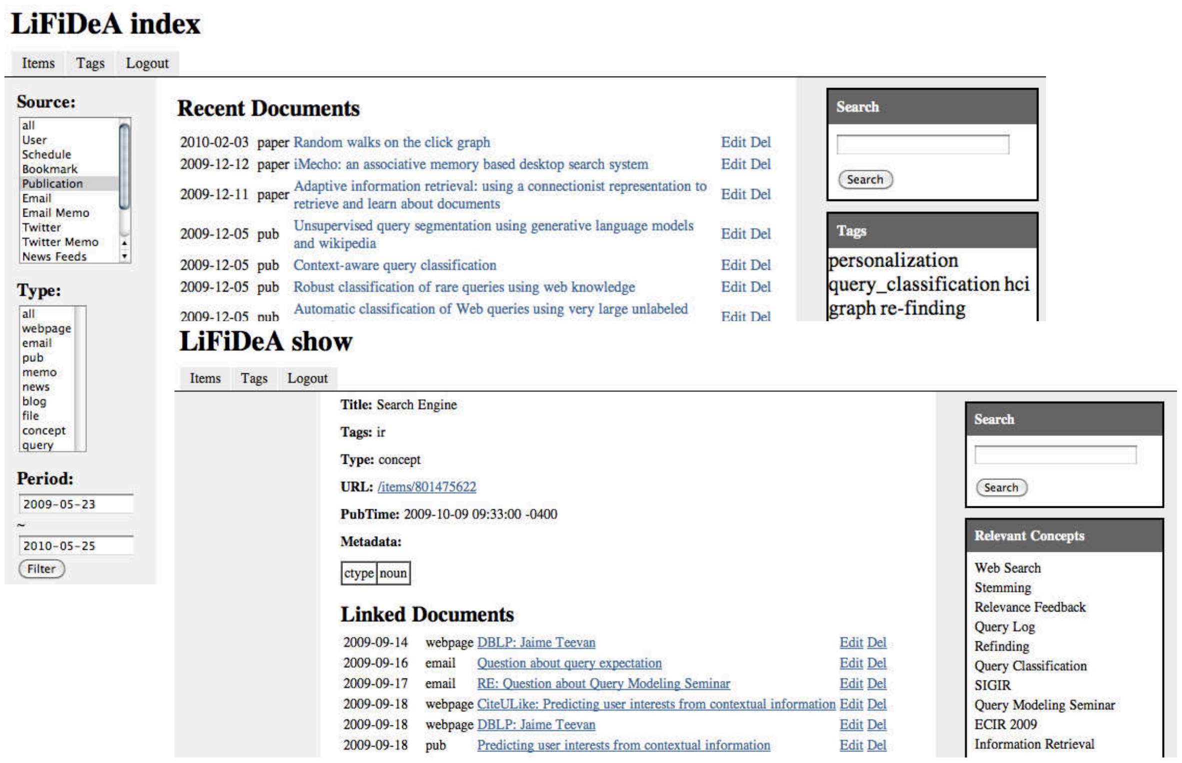
Given a collection of documents, the next step is to create concepts (e.g., names, domain terms, and so on), which constitute an extra layer one can use to browse documents. In LiFiDeA, concepts are *items* like documents, as described in Section 4.1.1. However, they are different in that the occurrences of concepts are extracted from documents, and that concepts are primarily used to access documents.

In addition to automatically extracting concepts from documents, the system allows the user to create concepts. There are several ways of adding concepts in LiFiDeA. A user can choose to promote a tag to a concept. The user can also decide to convert an appropriate document (e.g. Wikipedia article) to a concept. It is also possible to create a concept out of query words that the user types in to find documents.

4.1.3.3 User Interface for Browsing and Searching

The web interface shown in Figure 4.3 allows the user to browse the concept and document space. In the back, you can see a index page showing the list of publications along with tags. Here, users can perform full-text search by typing in keywords or faceted search by specifying conditions of filtering, which provides an starting point of browsing as depicted in Figure 4.2.

The front part of Figure 4.2 shows a page for a concept 'Search Engine'. Here, the documents mentioning the concept are listed as related documents, and related concepts are ranked by combining scores from each link types as features. When a user clicks on this ranked list of concepts, the system collects the user's clicks on relevant concepts and uses them as training data for adjusting feature weights.



LiFiDeA index

Items Tags Logout

Source:

all
User
Schedule
Bookmark
Publication
Email
Email Memo
Twitter
Twitter Memo
News Feeds

Type:

all
webpage
email
pub
memo
news
blog
file
concept
query

Period:

2009-05-23 ~ 2010-05-25

Filter

Recent Documents

2010-02-03	paper	Random walks on the click graph	Edit Del
2009-12-12	paper	iMecho: an associative memory based desktop search system	Edit Del
2009-12-11	paper	Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents	Edit Del
2009-12-05	pub	Unsupervised query segmentation using generative language models and wikipedia	Edit Del
2009-12-05	pub	Context-aware query classification	Edit Del
2009-12-05	pub	Robust classification of rare queries using web knowledge	Edit Del
2009-12-05	pub	Automatic classification of Web queries using very large unlabeled	Edit Del

Search

Search

Tags

personalization
query_classification hci
graph re-finding

LiFiDeA show

Items Tags Logout

Title: Search Engine

Tags: ir

Type: concept

URL: /items/801475622

PubTime: 2009-10-09 09:33:00 -0400

Metadata:

ctype	noun

Linked Documents

2009-09-14	webpage	DBLP: Jaime Teevan	Edit Del
2009-09-16	email	Question about query expectation	Edit Del
2009-09-17	email	RE: Question about Query Modeling Seminar	Edit Del
2009-09-18	webpage	CiteULike: Predicting user interests from contextual information	Edit Del
2009-09-18	webpage	DBLP: Jaime Teevan	Edit Del
2009-09-18	pub	Predicting user interests from contextual information	Edit Del

Search

Search

Relevant Concepts

Web Search
Stemming
Relevance Feedback
Query Log
Refinding
Query Classification
SIGIR
Query Modeling Seminar
ECIR 2009
Information Retrieval

Figure 4.3. LiFiDeA user interface. Back: Index page showing the list of publications along with tags. Front: The page of a concept 'Search Engine' showing related documents and concepts.

4.2 Ranking Suggestions for Browsing

A major challenge in implementing associative browsing is creating associations between items, which provide pathways for the user's navigation and therefore are critical for effective browsing. This is a particularly big obstacle in the domain of personal information where no ubiquitous mechanism exists for connecting information items, such as the hyperlink on the web.

In the data model we proposed, we can easily associate concepts and documents from which these concepts are extracted. However, creating associations between documents and between concepts is harder, since there is no single method that gives both high coverage and precision. This is where previous work turned to either manual annotations [44] [72] or a limited set of associations [16] [15] [29].

In this work, we propose a solution where the system's presentation of associations in the form of a ranked list is refined based on the user's feedback. In other words, we cast it as a similarity search problem and combine the values of many similarity metrics into a single score, by which the top k items are chosen as suggestions for browsing. In other words, our associative browsing model presents the user with a ranked list of related concepts or documents, generated by combining many measures of association with appropriate weights.

Another important task is finding appropriate weights for each feature, as we suggest the weighted combination of similarity measures for the ranking. We address this issue by using user's feedback. Given the space of concepts and documents, users can browse their personal information by navigating into related items. At the same time, users provide a stream of click feedback that is used to refine suggestions by the system.

In this section, we explain the features we used for representing an association between two items, followed by the methods we employed to learn feature weights. Since many features are similarity measures, we will use the term *similarity* interchangeably with *association*.

4.2.1 Features

The following subsections describe the features we used to rank suggestions for browsing. Note that some of features are applicable only for the ranking of concepts or documents. If that is the case, then the text in brackets after the name of the feature will reflect that.

4.2.1.1 Term Vector Similarity

We can create a term vector for each item based on the text in the title or content fields. Since many concepts do not have any text in their content fields, we use the documents in which the concepts occur. The term vector similarity score of two items is just the cosine similarity of the corresponding term vectors.

4.2.1.2 Tag Overlap

Since concepts and documents have tags associated with them, we can consider two items with common tags to be similar. Given two vectors of tags, we compute the tag overlap score using the cosine similarity.

$$TagSim(v_1, v_2) = \frac{overlap(v_1, v_2)}{\sqrt{size(v_1)} * \sqrt{size(v_2)}} \tag{4.1}$$

4.2.1.3 Temporal similarity

Intuitively, two items are deemed to be close to one another if the system indexes them within a short period of time, or if the user creates them within a short period of time. Therefore, the closer the creation of two items is in time, the higher their temporal similarity score. We got the feature value by taking the reciprocal of the difference in creation time (in seconds).

4.2.1.4 String Similarity (concept)

We compute the string-level similarity by dividing the Levenshtein distance between the titles of two concepts by the square root of the product of the title lengths as follows:

$$StringSim(s_1, s_2) = \frac{Levenshtein(s_1, s_2)}{\sqrt{size(s_1)} * \sqrt{size(s_2)}} \tag{4.2}$$

4.2.1.5 Co-occurrence (concept)

This feature counts how many times each concept pair occurs together in the collection's documents. It captures the semantic distance between two concepts. This metric is available only for the calculation of concept similarity.

4.2.1.6 Occurrence (concept)

This feature counts the number of times a concept has occurred in the document collection in log scale. Although all the other features measure some kind of similarity, this metric is intended to capture the popularity of a concept, since such concepts are likely to be clicked by a user.

4.2.1.7 Topical Similarity (document)

This feature relies on the topic model Latent Dirichlet Allocation (LDA) [11]. LDA is a hierarchical Bayesian model, which allows us to model a text document as a mixture of topics. To measure the similarity between two documents, we calculate the cosine similarity between the distribution of topics associated with each document. This is similar to computing the similarity of term vectors, except that each document is mapped to a vector of latent topics instead of terms.

4.2.1.8 Path / Type Similarity (document)

Since each document has a URI, we can compute a similarity score between two documents based on the *path*. Specifically, we calculate the similarity between two path strings by counting the word-level overlaps from the beginning of the path,

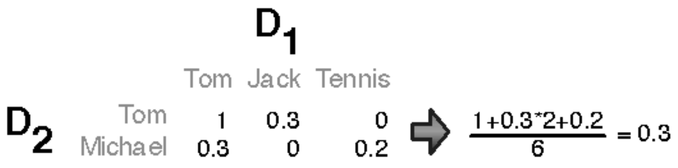


Figure 4.4. Calculation of concept overlap feature between two documents.

normalized by the number of words. Also, since each document has a *type* (e.g., email, pdf, etc.), we developed a binary feature based on whether two documents are of the same type.

4.2.1.9 Concept Overlap (document)

This feature is similar to tag overlap in that it considers two documents with common concepts to be similar. Unlike tags, since we can measure the strength of association between any two concepts, we can use it to measure the similarity between documents. In other words, even if two documents are linked to different sets of concepts, we can consider them to be similar if the concepts that each of them has are strongly associated.

As an example, Figure 4.4 illustrates how the value of the concept overlap feature is calculated. Although two documents share only one concept, the similarity between the concept pairs are calculated and then averaged to obtain the final concept overlap score.

4.2.2 Learning Feature Weights

One key step of our system is learning the weight of each feature. We examined two algorithms – iterative grid search and RankSVM [40].

4.2.2.1 Iterative Grid Search

In our problem setting, we need to find a weight for every feature. Since the number of our features is not too large, one reasonable approach is to vary one feature weight at a time to find the value that maximizes some metric of effectiveness. We then optimize the next parameter, fixing the values of the others, and repeat the whole procedure until we reach the convergence. As the target metric, we use the same metric (Mean Reciprocal Rank) as we use to evaluate the performance. To search for the optimum value for each parameter, we employ Golden Section Search[2].

4.2.2.2 Rank-learning Method

We also used RankSVM [40], which is a more sophisticated method designed for the rank-learning task. While we employed RankSVM because it accepts pairwise preferences between items as training data, any such learning method can be used here.

RankSVM is equivalent to SVM for classification except that the input data takes the form of pairwise preferences and the difference in feature values of two documents. In other words, instead of learning to classify between two sets of classes, the algorithm learns how to correctly predict the preference between two documents given as a set of feature differences.

4.2.2.3 Comparison of Learning Methods

The two learning methods used here have different characteristics. As for the objective function, grid search simply finds the set of parameters that maximizes the target metric, whereas the goal of RankSVM is to predict the pairwise preference relation with highest accuracy. There is another aspect in which the two methods differ. While grid search uses each click as a relevance judgment, RankSVM interprets each click as a pairwise preference. This difference will become more clear in the subsequent sections. We investigate the performance of the two learning methods in Section 6.4.3.

[2]http://economics.uwo.ca/faculty/klein/personal/numopt.pdf

4.3 Summary

In this chapter, we proposed an associative browsing model for personal information retrieval. We describe a data model and a prototype system that implements this model. We also show how the proposed model can be combined with keyword search for known-item finding task.

Our associative browsing model is composed of items (concepts and documents) and the links between them. Instead of displaying links of many types as they are, we generate a single ranked list of related items by combining the scores of many link types, which is then presented to the user. In calculating appropriate combination weights, we employ a learning framework which adjusts weights using click feedback from the user.

The evaluation for the proposed browsing model is described in the following chapters. In Section 5.4 and 5.5, we introduce evaluation methodology based on simulation and user study, respectively. In Section 6.4, we present evaluation results focusing on its role in known-item finding and the quality of suggestions for browsing.

CHAPTER 5
SIMULATION-BASED EVALUATION METHOD

So far in this thesis, we have focused on developing retrieval techniques for personal information retrieval (PIR). However, an equally important challenge is evaluating the retrieval techniques, since an effective evaluation would be a touchstone for further improvements. In fact, it has been argued that the research for PIR has been stagnant due to the challenges in evaluation [18] [42] [34].

Previously, most systems and techniques for PIR were evaluated by an instrumentation-based user study—deploying the system in a real environment and having it used by actual users. Although this kind of evaluation has its own benefits, it has several limitations. First, building a production-quality system requires considerable resources, and conducting a long-term user study is beyond time constraints of many research projects. Moreover, due to privacy concerns, the collections and usage logs from these studies are not open to other researchers. We will use the term 'naturalistic user study' to denote this instrumentation-based evaluation method.

As a result, although there were a number of previous attempts to build and evaluate PIR systems, there were not much in common among these efforts, and comparative evaluation among different methods was out of the question. To compare the statistics of documents gathered with the desktop collections used in previous research, we collected the data from publications or by contacting authors. Table 5.1 shows that desktop collections used in the past vary greatly in many aspects, such as the number of files and the composition of the collection in terms of file types.

Table 5.1. Statistics of desktop collections from previous research.

Previous Work	#Desktops	#Files	Query Length	Document Types
Dumais et al.[27]	225	36182	1.6	e-mails: 80% / documents: 10% / others: 10%
Chernov et al.[17]	14	3433	1.7	e-mails : 82.7% / documents : 17.3% / others: 0%
Cohen et al.[21]	19	N/A	N/A	e-mails: 0% / documents: 41.2% / others: 58.8%

Since the limitations of a naturalistic user study stem from employing actual users and their personal information, one way to address these issues is by using simulation techniques. Simulation in this context refers to replacing a component of evaluation with simulated equivalents, and we present simulation techniques for each component of PIR evaluation. Combining these techniques, we can automatically generate test collections for evaluating PIR methods. We also propose a user study method based on simulated tasks as opposed to naturally arising information needs.

Simulation-based evaluation is valuable for the evaluation of personal information retrieval for several reasons: First, since simulated evaluation can be done with considerably less time and efforts, we can get preliminary evaluation results for retrieval models before we perform an expensive user study. Furthermore, we can experiment with a variety of assumptions on user, task and the system by adjusting the parameters of simulation. This is an important aspect of evaluating a PIR method, because a PIR method should work for users with with a diversity of personal information and information needs.

However, given the nature of PIR, which aims at satisfying a user's information needs, it would be impossible for simulation techniques to completely replace studies with actual user involvement. In this thesis, instead of trying to replace user studies altogether, we propose using simulation as a component of a comprehensive evaluation framework. At the end of this chapter, we introduce an evaluation framework for PIR, where simulated evaluation methods are combined with realistic user studies based on different stages of a research project.

In what follows, we first describe major components and paradigms for PIR evaluation. We then describe our method for building simulated queries for the evaluation

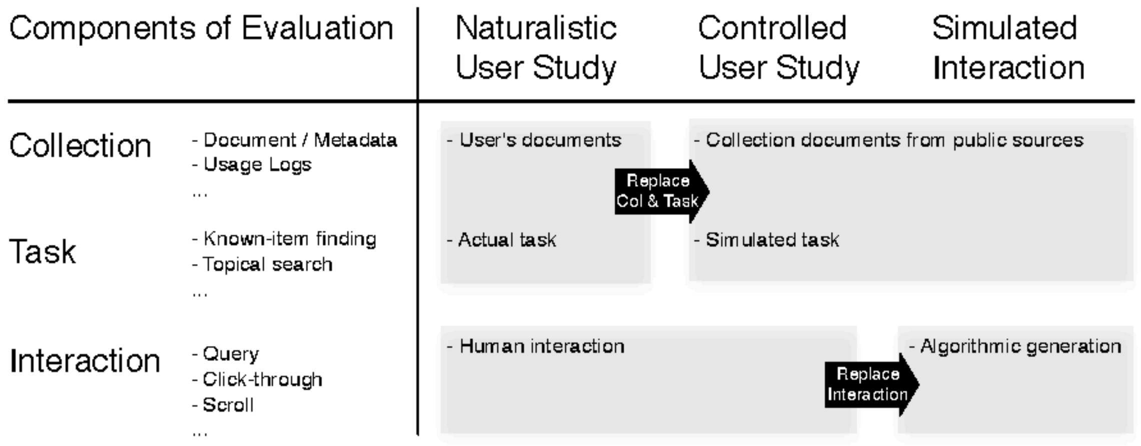


Figure 5.1. Components of PIR evaluation and the comparison of different evaluation strategies.

of term-based search, as well as a simulation technique for evaluating a scenario where users can use both search and browsing. We then introduce a user study method for collecting user interactions based on simulated tasks. Finally, we propose a evaluation framework that combines the simulated evaluation with user studies.

5.1 Evaluation Paradigms for Personal Information Retrieval

The evaluation of personal information retrieval, by definition, would require users with personal information, each of whom has a set of information needs to satisfy. Given the users, we can set up a PIR system we developed (or a retrieval method within an existing PIR system) on each user's machine, and capture a user's interaction with the system. This is a brief sketch of the type of naturalistic user study performed in previous studies.

Although this naturalistic user study is certainly realistic in that it is based on actual collections of personal information and tasks by real users, conducting this kind of study requires considerable investment in time and efforts. Also, performing the evaluation with real users assumes the existence of stable and working software, which would not be feasible at early stages of research. Finally, the resulting data from such naturalistic study can't be shared with other researchers for privacy issues.

Therefore, in our evaluation framework, we propose replacing each components of evaluation with simulated components, as shown in the left panel of Figure 5.1. The components include the collection of personal information, the retrieval task to be done within the system, and the user's interaction with the system.

By replacing actual collections and tasks with simulated ones, we can perform a user study where participants are asked to perform a set of tasks in a controlled environment. The DocTrack game presented in Section 5.5 is designed for conducting controlled user study based on known-item finding tasks. By simulating user interactions, we can eliminate human intervention from the evaluation altogether, where we use algorithmically-generated user interactions instead of the data captured from study participants. The proposed techniques for generating queries and user interactions enable such evaluation for known-item finding tasks as well.

In what follows, we introduce a set of techniques where we build a simulation of collection, task, and interaction in the context of the known-item finding task. The idea is to collect documents with similar characteristics to personal information, and then algorithmically generate user interactions, including queries and clicks on search results. User interactions generated can optionally be verified for their equivalence to existing user interaction data based on several techniques presented here.

5.2 Gathering Collection Documents

As a first step toward the simulated evaluation of PIR, we need a collection of documents that has the characteristics of personal information. The criteria that we used for the documents are that 1) the documents should be related to a particular person, 2) there should be of a variety of document types, 3) the different document types should have metadata or fields, 4) the collection should be of reasonable size, although there is no hard limit on size since real-world desktops vary considerably. The privacy of the target individual was another concern. In what follows, we describe two different ways of building a simulated collection of personal information.

5.2.1 Building Pseudo-desktop Collections

Given these constraints, the first method we employed is to re-use an existing collection for expert-finding task (TREC Enterprise collection [23]). Since the track had an expert-finding task, the list of people in W3C and their domains of expertise were available. We basically use the documents and the list of expertise related to these people to create pseudo-personal document collections.

Specifically, we first filtered the mailing list and webpage from the W3C collection to get documents that refer to each of target individuals. We then used a web search engine with the name, organization and specialization of each target individual as a query to find documents related to that person, repeating the procedure until gathered documents match the statistics of previously used desktop search collections. More details will be provided in Section 6.1.1.

In addition to satisfying the conditions above, this method provides a control over the types of collected documents since most search engines have the option to limit the search result by file type. Another advantage is that we can index rich metadata provided by a web search engine together with the documents. For the web search engine we used (Yahoo!), document title, URL, and summary were available.

5.2.2 Building the CS Collection

The procedure we described above provides a reasonable simulation of personal information, and is subsequently used for simulated interaction experiments we present in Section 6.3.2. However, in designing the controlled user study described in Section 5.5, we found that it would be more desirable to conduct the study with documents that participants are familiar with.

Since the study participants were recruited from the academic department the researcher belonged to, we collected public documents associated with the department, which would be familiar while raising no privacy concerns. Most of documents were found from the crawl of the department homepage, and we collected some emails from the department public mailing list. We called the resulting documents the computer science (CS) collection. More details will be provided in Section 6.1.2.

5.3 Generating Simulated Queries

Given the collection of documents, the next step is to create queries and corresponding relevance judgments. This is usually the most time-consuming part of building an IR test collection, because many documents need to be judged for relevance against a given query.

However, in this thesis, we present a set of techniques for generating simulated queries and relevant judgments automatically for known-item finding tasks. In this task, there exists a target item which a user wants to find using the retrieval system. The user's query is based on whatever the user remembers from the document. The proposed technique models this query generation process by choosing a target document and algorithmically selecting terms that would be used as a query.

In what follows, we present two query generation methods. The first method, introduced in previous work [7], became a foundation of the subsequent methods we proposed. The second method exploits the structured nature of personal information collection for term selection.

5.3.1 Document-Based Query Generation

Given a situation where a user is trying to find a document that she has seen (or created) previously, she may try to come up with whatever terms she can remember from the document. Based on this observation, Azzopardi et al. [7] suggested a set of methods for generating a known-item query by algorithmically selecting a set of terms from the target document, as illustrated below.

1. Initialize an empty query $q = ()$
2. Select document d_i to be the known-item with probability $P_{doc}(d_i)$
3. Select the query length s with probability $P_{length}(s)$
4. Repeat s times: $(k = [1..s])$
    4-1. Select the term t_k from document language model of d_i $P_{term}(t_k|d_i)$
    4-2. Add t_k to the query q
5. Record d_i and q to define a known-item/query pair

They suggested many parameterizations of P_{doc} and P_{term}, finding that inlink-based document selection improves the validity of the queries in general and that each collection requires different term-selection strategy.

5.3.2 Field-Based Query Generation

Previously, Azzopardi et al. [7] showed that the generated queries can be used for retrieval experiments in a web environment. However, a typical personal information collection has different characteristics Specially, as we describe in Section 3.2.2, we assume that the users' querying behavior would be different for personal information retrieval because each document is composed of multiple fields.

Therefore, we modified their query generation method for PIR by incorporating the selection of fields in the generation process, which results in the following algorithm:

1. Initialize an empty query $q = ()$
2. Select document d_i to be the known-item with probability $P_{doc}(d_i)$

3. Select the query length s with probability $P_{length}(s)$

4. Repeat s times: $(k = [1..s])$

    4-1. Select the field $f_j \in d_i$ with probability $P_{field}(f_j)$

    4-2. Select the term t_k from field language model of f_j $P_{term}(t_k|f_j)$

    4-3. Add t_k to the query q

5. Record d_i and q to define a known-item/query pair

The modification here is step 4., where we choose the field from which the query term is selected. Our hypothesis is that users may (implicitly) choose fields when they choose query terms, which has an intuitive appeal given that some document fields (e.g., *To* and *From* in email) are very important in characterizing the document. In Section 5.3.3, we verify this hypothesis by showing that field-based query generation method creates queries that are more similar to actual user-generated queries than the document-based generation method.

Note here that we only use terms in the target document, which may be an unrealistic model. However, the issues with the validity of the generated queries are reduced when they are used solely for comparative evaluation of retrieval methods, since all methods use the same set of queries. It would be possible to include terms outside the document in many ways, for instance by interpolating P_{term} with a collection language model. We leave the investigation of such methods to future work.

Although there can be many variations in choosing P_{doc} and P_{field}, we use a uniform distribution that assigns the same probability for every available document and field, respectively. For P_{length}, we use the statistics of previously used desktop collections. For P_{term}, we use uniform selection, TF-based selection, IDF-based selection and TF*IDF-based selection, as suggested in Azzopardi et al. [7]

5.3.3 Verifying Generated Queries

In general, any simulation method would be meaningful only if it can closely reflect the phenomenon it tries to model. For the retrieval experiments using the generated queries to be meaningful, we need to show that they are equivalent in some sense to hand-built queries.

To do this, past work [7] introduced the notions of predictive and replicative validity, and experimented with replicative validity. Predictive validity means whether the data (e.g., query terms) produced by the model is similar to real queries, while replicative validity indicates the similarity in terms of the output (e.g., retrieval scores).

In this thesis, we experiment with both predictive and replicative validity as they address different aspects of the query generation technique. Predictive validity is verified by comparing query terms and therefore is independent of the retrieval method. In contrast, replicative validity compares the distribution of scores returned by the system and is accordingly dependent on the choice of retrieval method.

Another point is that while the verification of predictive validity does not involve randomness once P_{term} is given, the same does not hold true for replicative validity. This is because the query generation procedure in general involves random selection of query terms, which in turn changes the distribution of scores. In sum, these two measures are complementary in that predictive validity is more stable, while replicative validity is more strongly related to our eventual goal (retrieval results).

5.3.3.1 Predictive Validity

In verifying predictive validity, we need to evaluate how close the generated queries are to hand-built queries. To accomplish this, since query generation involves the choice of term distribution P_{term}, we suggest using the generation likelihood $P_{term}(Q)$ of the manual query Q. This can be computed with the term distribution P_{term} from the given query generation method, as follows:

$$P_{term}(Q) = \prod_{q_i \in Q} P_{term}(q_i) \tag{5.1}$$

Getting P_{term} for document-based query generation method is straightforward since we can just use the simple maximum-likelihood estimates for each word. For the field-based query generation method, since every field has different P_{term}, we need to take the linear interpolation of P_{term} for all fields. Since we use a uniform probability for field selection, P_{term} for each field can be combined with equal weights.

5.3.3.2 Replicative Validity

Verifying replicative validity involves comparing some outcome of the simulation method and the target of the simulation. In our case, the goal of our simulation is to generate known-item query and relevant document pairs, and together they can generate a score for a particular retrieval system. Therefore, a natural comparison is between the distribution of scores from generated queries and hand-built queries.

In previous work, Azzopardi et al. [7] measured replicative validity by the two-sided Kolmogorov-Smirnov test (KS-test) using the score samples of real and generated queries as input. The KS-test is an independent two-sample test which tests the null hypothesis that the two samples may come from the same distribution and the result is sensitive to both the location and the shape of the samples. Since the KS-test quantifies the similarity between the empirical distribution functions of two samples, we can conclude that two distributions are equivalent if the resulting p-value is greater than a certain threshold.

5.4 Generating Simulated User Interactions

So far we have looked at methods for generating known-item finding queries. While the query generation method enables the evaluation of keyword search methods, it does not generalize into other modes of information access, including the associative

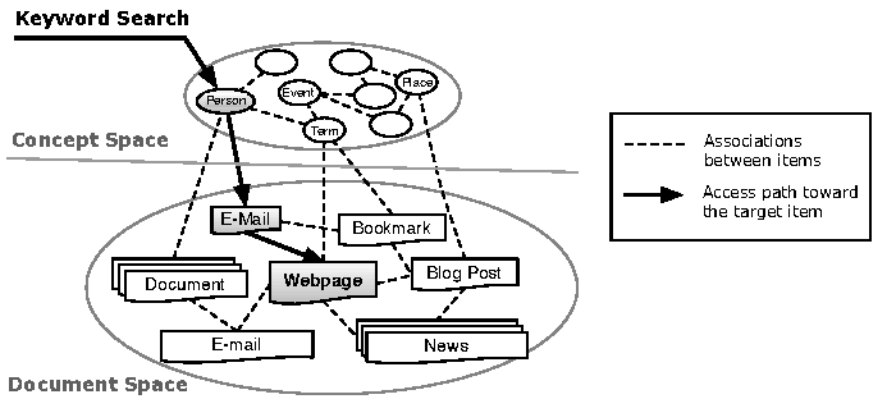


Figure 5.2. An illustration of the suggested associative browsing model. Dotted lines represent the association between documents and concepts. Directed lines denote how a user can access the target webpage by using keyword search and associative browsing.

browsing model introduced in Chapter 4. We somehow need to simulate other types of user interaction.

In this section, we extend the notion of query generation to a general user simulation from which we can generate arbitrary user input. In particular, our goal is to build a simulation of the actual user behavior for the evaluation of the whole retrieval framework, including multiple modes of information access such as keyword search and associative browsing. The user model is parameterized to simulate various aspects of the user, including the level of knowledge and the behavioral pattern.

The user model follows the pattern of interaction defined in the scenario of known-item finding described in Section 4.1.2. Figure 5.2 provides an example of information seeking where the user combine search and browsing. Here, the user first find the concept of a person name, and then browse into a email mentioning the name, finally reaching the target webpage related to the email.

We also can represent such process using a state diagram. Figure 5.3 shows a diagram of state transitions that are involved in the sequence of interactions between the user and the system. Here, each state represents a particular stage in the user's

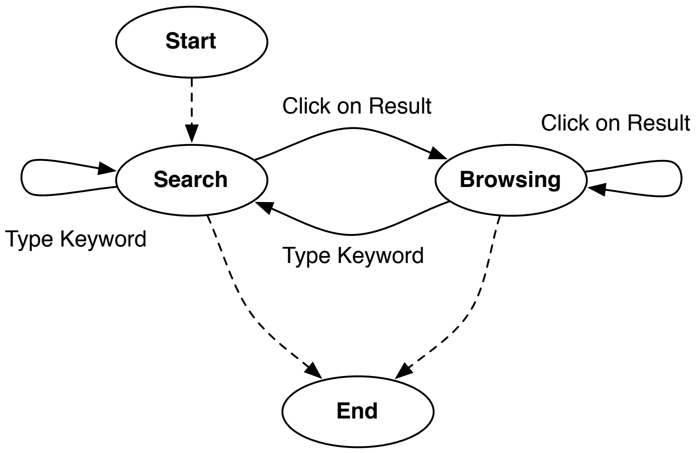


Figure 5.3. A state transition diagram for suggested probabilistic user model.

information seeking, and the arrows denote user actions that lead to the transition between states.

As a starting point, we expect the user to perform a keyword search using the terms he remembers from the document. If the initial search is successful, he can finish the session. Otherwise, he can either reformulate the query or click on one of the top documents to browse into related items. This process continues until he finds the target document or he reaches the limit of his patience.

We divided the model into three components — keyword search, associative browsing, and the transitions between states. The keyword search component models how the user would choose terms for search, and the associative browsing component is responsible for modeling the user's clicks on the ranked list. The state transition part is concerned with the decision made by the user on whether to use search or browsing, or whether to continue the current session or terminate. In what follows, we explain each component in detail.

5.4.1 Keyword Search Model

We previously introduced techniques for generating known-item queries that can be used for PIR experimentss. Since we are dealing with the known-item finding task in a personal documents collection, we used the query generation model suggested in [46] to get queries targeted for finding a document.

As a keyword search model, we employ the field-based query generation method introduced in Section 5.3.2. More specifically, given a target document and pre-specified length of query, we choose each query-term from a term distribution P_{term} estimated from the document until we reach the limit in pre-specified length.

5.4.2 Associative Browsing Model

In the scenario of known-item finding we assume in this thesis, associative browsing is used as an alternative to keyword search. When a keyword query returns only marginally relevant results, a user will click on one of top retrieved documents to browse related documents. By following this chain of associations, one can locate the target document.

A central modeling target for browsing behavior is the user's clicks on retrieved results, where we have several choices in modeling this behavior. The first choice in modeling browsing is the level of knowledge the user has about the collection and target documents. A more knowledgable user will make a better choice in deciding which document to click on. In this work, we employ three levels of user's knowledge — *random*, *informed* and *oracle*, which correspond to the status of no knowledge, partial knowledge and complete knowledge, respectively.

To implement the level of knowledge in user clicks on the ranked list, we need to evaluate the candidate documents in terms of their value in retrieving the target document. In the known-item finding task, the target document is known and each click to a candidate document leads to a ranked list which may contain the target document at some position. Therefore, we can use a retrieval effectiveness measure (MRR) for each candidate document to evaluate its value in locating the target document. And this estimation of candidates' value provides a ground for modeling user's knowledge.

Specifically, while the *random* user may click on a random position of a ranked list, the *informed* user will choose documents from the distribution of candidate documents whose probability corresponds to the estimated value of each candidate document. Finally, the *oracle* user will always click on the document with highest value. The behavior of the *oracle* user is greedy in that the choice is based on what seems the best each moment, and we show in Section 6.4.2.2 that this greedy strategy does not always lead to the highest success.

Another interesting parameter in user modeling is the variations in browsing behavior — how many documents are visited at each time the user sees a ranked list, and in what order. Although there are many possibilities in modeling user's browsing behavior, we employed two browsing strategies introduced in Smucker et al. [78] — depth-first and breadth-first.

Figure 5.4 illustrates three examples of browsing strategies, where each node represents a document, and the number in each dot represents the order in which documents are visited. Since each document corresponds to the ranked list of related documents, each arc in the figure corresponds to a user's click. You can see how the fan-out and the browsing order corresponds to the browsing styles of different users.

In sum, two parameters we use in modeling users' browsing behavior are the fan-out (how many documents the user clicks on a ranked list), and the browsing strategy employed (breadth-first search and depth-first search). Here, higher fan-out means more exploration than exploitation (more clicks per ranked list), while BFS and DFS represents exploration-first strategy and exploitation-first strategy, respectively.

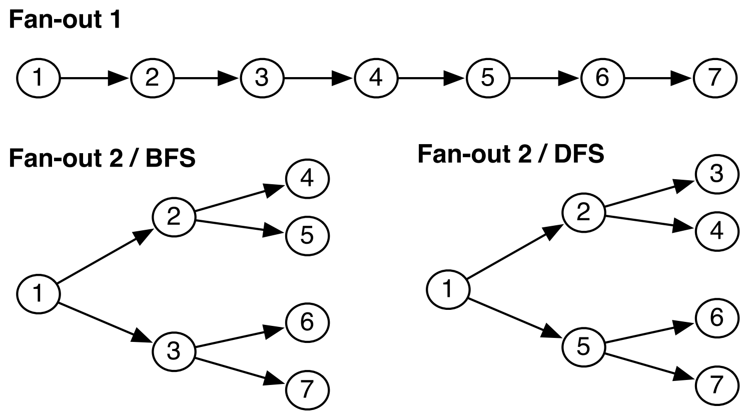


Figure 5.4. An illustration of two browsing strategies: breadth-first search (BFS) and depth-first search (DFS), both with fan-out of 2. Numbers represent the order of documents in which they are visited.

5.4.3 State Transition Model

The rest of the simulated user model is concerned with the decision made by a user on whether to use search or browsing, or whether to continue the current session or terminate. There can be many factors that can affect the user's decision, including whether they can come up with effective search terms, the perceived quality of results from search or browsing, and so on.

Since our main goal in this work is to evaluate the role of browsing as a complement for search, we used a simplifying assumption that users would choose to browse if the initial search is only marginally successful. Here, the technique for measuring the effectiveness of a ranked list using pre-specified target document can be used to model the transition from one method to the other. Although there can be many other considerations in modeling this component, we leave them for future work.

5.5 Collecting User Interactions using a Game

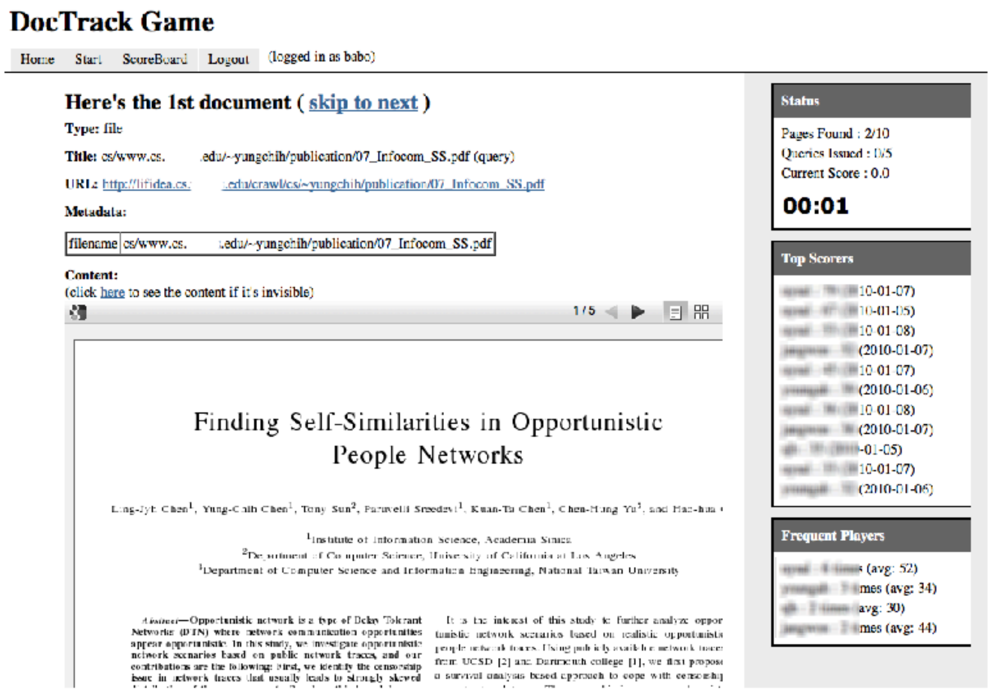
Evaluation based on simulated interactions as introduced above can be valuable in studying the characteristics of different retrieval methods. However, such a simulated user interaction cannot be a substitute for a human data, because the validation techniques concerns only a specific aspect of data, and even such validation requires human log data. As an alternative way of evaluating personal information retrieval (PIR) with minimal human involvement, we suggest a game-based evaluation method in which participants are asked to find a set of target items in a competitive setting.

This game-based user study has several benefits compared to a traditional user study. First, it induce higher motivation among participants thanks to a more interesting task and the competitive nature of the game. Secondly, a tight experimental control is possible since participants are asked to complete a set of given tasks under constraints provided by the game designer. Reusability of data is another benefit, because public documents are used and most participants are willing to make public their activity logs. Last but not least, developing and running a game-based user study can be done within a relatively small amount of time and effort.

One can see that a game-based user study is not without issues, which stems from its artificial nature. The situation we created within the game is not the same as actual search tasks, and the competitive environment may lead to unrealistic behaviors. Lastly, it does not use personal information or actual search task. However, we believe that these can be minimized by sensible design and execution of the study, which we will illustrate using our studies as examples.

There has been several attempts to introduce the game-like user interface of some kind for the evaluation of a IR system, as reviewed in Section 2.7. By adapting the PageHunt game [61] to our problem domain, we developed the DocTrack game [47] for evaluating PIR methods in the context of known-item finding, as shown in Figure 5.5. We made several modifications to the original PageHunt game:

First, since people generally have good knowledge of their own desktops, we collected documents that the participants are familiar with, and let each of them browse the collection for some time before starting the game. Second, to simulate a typical



DocTrack Game

Home Start ScoreBoard Logout (logged in as babo)

Here's the 1st document (skip to next)

Type: file

Title: cs/www.cs. .edu/~yungchih/publication/07_Infocom_SS.pdf (query)

URL: http://lifidea.cs. .edu/crawl/cs/~yungchih/publication/07_Infocom_SS.pdf

Metadata:

filename	cs/www.cs. .edu/~yungchih/publication/07_Infocom_SS.pdf

Content:
(click here to see the content if it's invisible)

1 / 5

Finding Self-Similarities in Opportunistic People Networks

Ling-Jyh Chen[1], Yung-Chih Chen[1], Tony Sun[2], Paruvelli Sreedevi[1], Kuan-Ta Chen[1], Chen-Hung Yu[3], and Hao-hua

[1]Institute of Information Science, Academia Sinica
[2]Department of Computer Science, University of California at Los Angeles
[3]Department of Computer Science and Information Engineering, National Taiwan University

Abstract—Opportunistic network is a type of Delay Tolerant Networks (DTN) where network communication opportunities appear opportunistic. In this study, we investigate opportunistic network scenarios based on public network traces, and our contributions are the following: First, we identify the censorship issue in network traces that usually leads to strongly skewed

It is the interest of this study to further analyze opportunistic network scenarios based on realistic opportunistic people network traces. Using publicly available network traces from UCSD [2] and Dartmouth college [1], we first propose a survival analysis based approach to cope with censorship

Status

Pages Found : 2/10
Queries Issued : 0/5
Current Score : 0.0

00:01

Top Scorers

[redacted]10-01-07)
[redacted]10-01-05)
[redacted]10-01-08)
[redacted] (2010-01-07)
[redacted]10-01-07)
[redacted] (2010-01-06)
[redacted]10-01-08)
[redacted] (2010-01-07)
[redacted]-01-05)
[redacted]10-01-07)
[redacted] (2010-01-06)

Frequent Players

[redacted]s (avg: 52)
[redacted]mes (avg: 34)
[redacted]avg: 30)
[redacted]mes (avg: 44)

Figure 5.5. The screenshot of the DocTrack Search game. The user is being shown a document.

known-item search scenario, we showed participants multiple documents and asked them to find one of them without specifying which one is the target document. Third, we used a document viewer that can show documents of any types (e.g. pdf, doc, and ppt) in the same way they are seen on the desktop.

Figure 5.6 shows the flow of the DocTrack game. As a starting point, the system shows two candidate documents to the users for a certain period of time, and then randomly chooses one target document. The user then combines keyword search and associative browsing to find the item. Each keyword query or click on the ranked list is considered a trial, and the user is given 10 trials for each session. The score is determined by the rank position of each target document in the final rank list—the higher the position, the higher the score.

The rationale behind the presentation of multiple target documents is to simulate the state of vague memory for the target document. For instance, if the user is shown

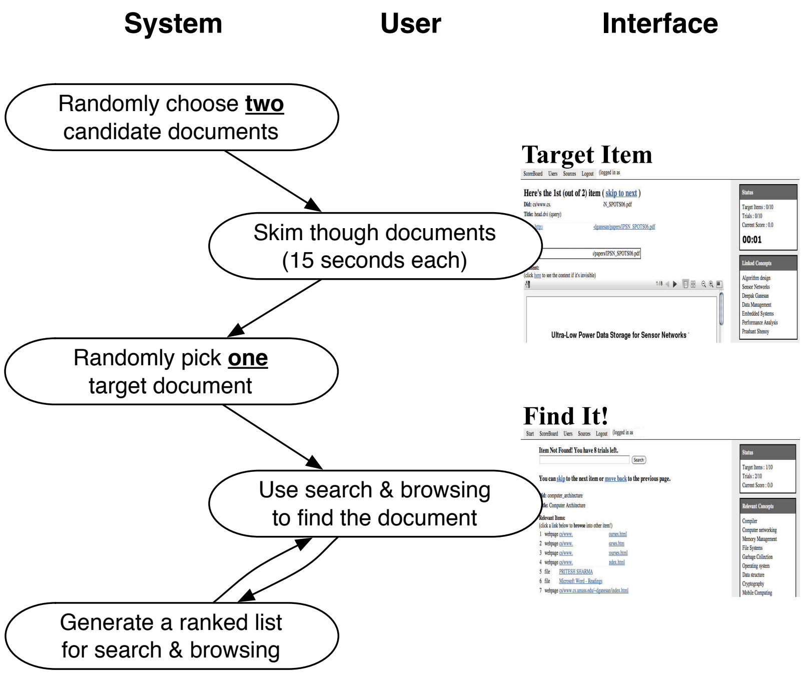

System
User
Interface
Randomly choose **two** candidate documents
Skim though documents (15 seconds each)
Randomly pick **one** target document
Use search & browsing to find the document
Generate a ranked list for search & browsing
Target Item
Find It!

Figure 5.6. The scenario of DocTrack search and browsing game.

an *email* and a *webpage* sequentially and then asked to find one of them, he or she might get confused about the content of the two documents. We assumed that this kind of confusion would be similar to the memory of a typical known-item searcher. We leave it as future work to verify whether this kind of trick realistically simulates the state of memory for known-item finding.

We ask each user to find a total of 10 items. The score is determined by the location of each target document in the final rank list—the higher the rank, the higher the score. In what follows, we introduce two variants of the game that are designed to evaluate different interaction methods.

5.5.1 DocTrack Search Game

The DocTrack search game is designed for collecting interaction data based on keyword search. The DocTrack search game follows the scenario shown in Figure

5.6, except that it allows only keyword search for finding the target document. We collected the search log data used for experiments reported in Section 6.3.2.2 using the game.

Compared to the method of collecting queries described in Section 6.1.1, using the DocTrack game, we could gather a large quantity of realistic queries together with the whole session log data. This in turn allows us to train discriminative learning models, which typically requires a large amount of training data. We report on how the data was used to train the type prediction model in Section 6.3.2.2.

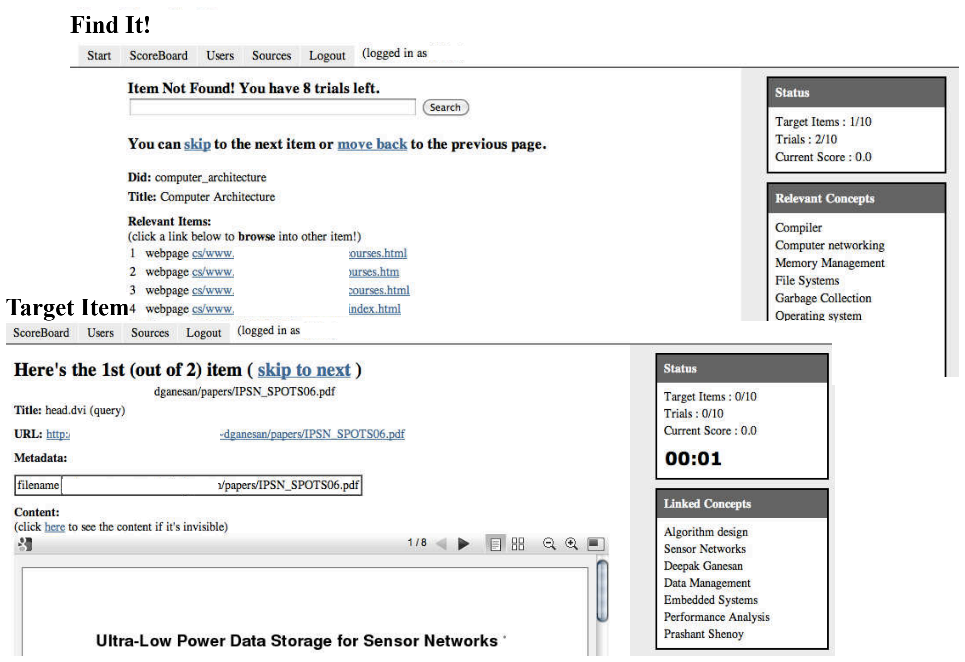
5.5.2 DocTrack Search & Browsing Game

For evaluating the associative browsing model introduced in Chapter 4, we designed the DocTrack search and browsing game. As shown in Figure 5.6, the participants can find a target document by combining keyword search and associative browsing. Figure 5.7 (front) shows the interface that people used to look through the initial set of documents. The screenshot in the back of Figure 5.7 displays the interface that people used to find the target document.

We ran two rounds of user studies with slightly different settings. In the first round, users were asked to find the target document using only search and browsing of documents. In other words, they did not have access to the concepts. In the second round, concepts were available for searching and browsing, thereby providing full access to the model. The rationale behind this two-stage design is to evaluate the role of each system component and to help users gradually familiarize themselves with the system. More details of the experiments can be found in Section 6.4.1.

5.5.3 Lessons Learned

We learned several lessons during our studies. First of all, we found out that it is important to explain the method of gameplay since users can play games online at a convenient location and time, whereas many user studies happen on-site. Especially


Find It!

(logged in as

Item Not Found! You have 8 trials left.

You can skip to the next item or move back to the previous page.

Did: computer_architecture
Title: Computer Architecture

Relevant Items:
(click a link below to **browse** into other item!)

1 webpage cs/www. ourses.html
2 webpage cs/www. ourses.htm
3 webpage cs/www. courses.html
4 webpage cs/www. index.html

Status
Target Items : 1/10
Trials : 2/10
Current Score : 0.0

Relevant Concepts
Compiler
Computer networking
Memory Management
File Systems
Garbage Collection
Operating system

Target Item

(logged in as

Here's the 1st (out of 2) item (skip to next)
dganesan/papers/IPSN_SPOTS06.pdf

Title: head.dvi (query)
URL: http:/ -dganesan/papers/IPSN_SPOTS06.pdf
Metadata:

filename	ı/papers/IPSN_SPOTS06.pdf

Content:
(click here to see the content if it's invisible)

Ultra-Low Power Data Storage for Sensor Networks

Status
Target Items : 0/10
Trials : 0/10
Current Score : 0.0
00:01

Linked Concepts
Algorithm design
Sensor Networks
Deepak Ganesan
Data Management
Embedded Systems
Performance Analysis
Prashant Shenoy

Figure 5.7. The screenshot of the DocTrack Search and Browse game. Back: the user interface for finding a target document by searching and browsing. Front: a target document is being shown along with related concepts.

for our second study, since it involved a new method of finding the target documents, we prepared a screencast explaining the functionality of the game interface in addition to a couple of examples, which got positive reactions from the participants.

We also found that special attention is required to ensure that playing the game is equivalent to using the system in a natural setting. Although one advantage of game-based evaluation is that users are motivated to get higher scores than other users, this competitive environment sometimes led to behavior that would not usually happen in a natural setting, such as memorizing the whole title of the document. We adjusted the scoring scheme of each search session so that users are discouraged from typing in many keywords.

Lastly, keeping users interested in playing throughout the whole game is another important factor for the successful data collection. One of the issues we had is that users would randomly click on the ranked list when faced with items they are not familiar with. The problem here is that the training data we extract from their clicks will be noisy and the ranked list presented to them will not be of a very good quality. We solved this issue by allowing users to skip to the next target item whenever they feel like.

5.6 Three-stage Evaluation Methodology

In synthesizing different evaluation approaches introduced in this thesis as well as previous work, we present a evaluation methodology based on different stages of a research project, where each stage is designed to verify and refine research ideas with the different level of progress.

5.6.1 Stage 1: Simulated Interaction

At an early stage of project, researchers would have only rough hypotheses about the problem, with no facility (e.g., prototype software) to verify their ideas. Simulated interaction will be useful at this stage by guiding the implementation of the retrieval method, and allowing them to make rough estimates on the relative performance of different algorithms [46].

While simulated interaction in itself may not be sufficient for the final validation of research ideas, it can be a first step by which initial hypotheses are verified and the experimental infrastructures are prepared. If an initial hypothesis can be falsified based on simulation results at this stage, further investment can be prevented. For instance, an experimental retrieval algorithm may shown to underperform existing methods significantly, and the plans for a user study can be delayed temporarily.

5.6.2 Stage 2: Controlled User Study

With the initial validation of research ideas through Stage 1, researchers can perform a user study using simulated tasks using the simulated collection developed in Stage 1. For instance, the game-based user study described in Section 5.5 can be used here. In this stage, a prototype user interface can be built and checked for usability, and various system parameters can be tuned based on log data.

User studies of this kind are less costly than diary studies since they do not require client-side instrumentation and can be done within a short time. Another benefit is the possibility of sharing the log data, since they do not use any private information. However, sometimes its artificial nature becomes problematic, which requires the naturalistic user study described in what follows.

5.6.3 Stage 3: Naturalistic User Study

While the evaluation method at Stage 2 can be used to perform evaluations with reasonable human involvement, it is inadequate for some classes of research problems because aspects of the collection and task are hard to model. For instance, evaluating retrieval methods which exploits the user's task context would require actual user involvement.

In this case, a long-term user study which involves the instrumentation of software to users' system may be required. Our suggestion is not to eliminate this kind of user study completely, but to avoid it when simulation techniques can be an alternative. After two stages of simulation studies, many factors that can potentially cause problems for user study would have been eliminated, which would greatly increase the chances for success.

5.7 Summary

In this chapter, we first described several evaluation paradigms for personal information retrieval, and then introduced a set of methods for collecting documents, generating queries and validating them. We also presented how the complex sequence of interactions can be generated by combining the keyword search model with the browsing model.

We also proposed using a game-based user study for evaluating personal information retrieval. We designed two types of games for evaluating field-based search and associative browsing, respectively. Our experiments in Chapter 6 show that it is possible to collect a large amount of log data from many participants, which then can be used to train and evaluate both the search and browsing models we proposed.

Finally, we introduced a three-stage evaluation framework for personal information. We described how the simulated evaluation and naturalistic user study can be sensibly combined throughout the progress of a research project.

The proposed techniques are already adopted by PIR research community. Recently, we organized a workshop[1] based on the collections we created with these methods. Also, several recent work [33] [38] suggested extensions to the methods introduced here.

[1]Evaluating Personal Search Workshop in ECIR'11

CHAPTER 6

EVALUATION RESULTS

In this chapter we present the experimental results for the retrieval and evaluation methods we have introduced. We first describe several simulated test collections we built using the methods introduced in Chapter 5, because these collections are extensively used for the following experiments.

For the field-based search models (Chapter 3), a main goal of the experiments is to evaluate the retrieval and type prediction methods against state-of-the-art baselines. To ensure a comprehensive evaluation of the proposed methods, we used several well-known structured document collections as well as the simulated test collections we built. We also analyze the performance characteristics of field-based retrieval models from the perspective of field relevance (or field weighting, equivalently).

For the associative browsing model (Chapter 4), we aim to evaluate its value in the known-item finding task and the quality of browsing suggestions. We first evaluate its ability to complement keyword search in the test collections we created, based on both a user study and simulation results. Since we introduced a learning framework for ranking candidates for associative browsing, we also present experimental results for the quality of the ranking.

In what follows, we first describe the simulated test collections we built along with several validation results. We then present the evaluation of retrieval methods we proposed earlier—the field-based search models and the associative browsing model.

Table 6.1. Number and average length of documents for pseudo-desktop collections.

Type	Jack		Tom		Kate	
email	6067	(555)	6930	(558)	1669	(935)
html	953	(3554)	950	(3098)	957	(3995)
pdf	1025	(8024)	1008	(8699)	1004	(10278)
doc	938	(6394)	984	(7374)	940	(7828)
ppt	905	(1808)	911	(1801)	729	(1859)

6.1 Simulated Test Collections

In this section, we describe two test collections we built for evaluating personal information retrieval. As we introduced the methodology used for building these collections in Section 5.2, we focus on describing the collections themselves here.

6.1.1 Pseudo-desktop Collections

Based on the method described in Section 5.2.1, we built three pseudo-desktops collections so that they contain typical file types in a desktop such as *emails*, webpages (*html*) and office documents (*pdf*, *doc* and *ppt*) related to each of three specific individuals.

To get the emails related to a person, we filtered the W3C[1] mailing list collection where the name occurrences of each person were tagged [8], which enabled us to identify several individuals whose activities in W3C were prominent. For other document types, we collected up to 1,000 documents for each individual and document type, using the Yahoo! search API with the combination of name, organization and speciality of each pseudo-user as query words. In identifying the specialty of each individual, we used a list provided by the TREC Enterprise Track [23].

Table 6.1 lists the statistics from the resulting pseudo-desktop collections corresponding to three pseudo-users — "Jack", "Tom" and "Kate". Although these are

[1]W3C is an international organization which focuses on numerous standardization efforts regarding the internet.

Table 6.2. Example queries from TREC Enterprise track and pseudo-desktop collections.

TREC Manual Queries
Preliminary Report from the WSDL Attributes Task Force
XML DSig 99
MobiQuitous 2004 latest deadlines
Pseudo-desktop Manual Queries
Martyn Jan OCLC
proof checking
syntax for RDF
Pseudo-desktop Generated Queries
jose 03 kahan
john descendent boy
connolly sat

prominent figures in W3C and all the collected documents are publicly available, we have anonymized their names for privacy reasons. Table 6.2 shows example queries from the TREC Enterprise Track, as well as the manual and generated queries from the pseudo-desktop collections.

6.1.2 CS Collection

We created the CS collection by gathering public documents from the Computer Science department of the University of Massachusetts Amherst. Following the methodology introduced in Section 5.5, we collected emails from the department mailing list, news articles and blog postings on technology, calendar items of department announcements, webpages and office documents crawled from the department and lab websites.

Table 6.3 shows the overall statistics of the collection. For all document types, *title* and *content* fields were indexed. There were also type-specific fields such as *date*, *sender* and *receiver* for email, *tag* and *author* for news articles, *starttime* and *location* for calendar items, *URL* for webpages, and *filename* for office documents.

We had 20 participants who were students, faculty members and staffs in the department, all moderately familiar with the documents in the collection. In total,

Table 6.3. Number and average length of documents in a computer science (CS) collection.

Type	#Docs	Length
email	851	731
news article	170	352
calendar item	354	306
webpage	4727	539
office document	1887	357

Table 6.4. Query examples with corresponding target collections for a CS collection.

Query	Target Collection
reminder jeffrey johns	email
2010 candidate weekend	calendar item
yanlei xml dissemination	office document
cs646 homework html	html

66 DocTrack games were played and 984 queries were collected using 882 target documents, some of which are shown in Table 6.4.

The average length of queries was 3.97, which is longer than the reported length (2) in other desktop search studies [30]. This may be due to people paying more attention to the task in the competitive game setting compared to a typical desktop search. The standard deviation (1.85) of the query length was also quite high, implying that there was a considerable variation among the querying behavior of individuals.

6.2 Validating Generated Queries

We now present validation results for generated queries based on the techniques described in Section 5.3.3. Generated queries are verified in terms of predictive and replicative validity using some of the retrieval methods described in Section 3.2.1. Since the validation methods require a set of queries formulated by humans, we experiment with TREC Enterprise track queries, where we have 150 known-item finding queries with target documents.

Table 6.5. Sum of generation likelihood (in log) for different generation methods on TREC queries. Field-based generation method with TF*IDF-based term selection show highest likelihood.

Extent : Document		Extent : Field	
P_{term}	$\sum_{q \in Q} P^D_{term}(q)$	P_{term}	$\sum_{q \in Q} P^F_{term}(q)$
Uniform	-26.457	Uniform	-23.460
TF	-22.782	TF	-22.394
IDF	-21.876	IDF	-17.990
TF*IDF	-18.269	TF*IDF	-17.180

6.2.1 TREC Collection

We first report the verification result of predictive validity using the generation likelihoods of 150 TREC queries. As we discussed in Section 5.3.3.1, we aggregated the likelihood that each manual query $q \in Q$ would be generated from the term distribution P_{term} of each generation method. Table 6.5 shows that field-based query generation methods have higher likelihoods of generating manual queries than document-based generation methods. Although it does not provide an absolute criteria for judging the equivalence to TREC queries, it shows that the term distributions for field-based generation methods are relatively similar to manual queries. Among the term selection methods, term selection by TF*IDF was shown to have a higher probability for generating TREC queries, reflecting people's behavior of choosing popular and discriminative terms.

We then evaluated each query generation method in terms of the replicative validity using the Kolmogorov-Smirnov (KS) test as discussed in Section 5.3.3.2. The results are in Table 6.6. Each cell contains the average p-value of the KS-test for the score distributions of TREC queries and generated queries using the corresponding retrieval method. Here, the values greater than 0.05 means that we cannot reject the null hypothesis that the score distribution from the generated queries and manual queries are indistinguishable, thereby providing some evidence on the equivalence in the output of generated and manual queries.

We experimented with Document Query-Likelihood (DQL) [66], BM25F, the Mixture of Field Language Models (MFLM), the Probabilistic Retrieval Model for Semi-structured data (PRM-S) and the Field Relevance Model (FRM). Here we report only the results of DQL, PRM-S and PRM-D because they were the retrieval models for which replicative validity was established for any of the query-generation methods. The result shows that the field-based query generation methods have higher p-values in general, and the uniform or TF-based term selection methods have p-values greater than the threshold (0.05) for all retrieval models tested.

The verification results for predictive and replicative validity do not completely agree in the sense that the generation method with highest predictive validity (field-based generation with TF*IDF term selection) does not have the highest replicative validity for the retrieval models we experimented with. As mentioned before, a possible cause is that the replicative validity is verified by comparing score distributions, which can be affected by many factors other than the collection statistics of query terms.

The results in Table 6.6 support this explanation since the field-based generation method with the TF*IDF term selection has the highest replicative validity for DQL yet fails on PRM-S and PRM-D, which incorporate the mapping probability into the field-level query likelihood score. This is also consistent with the results of Azzopardi et al. [7] where TF*IDF term selection method was shown to have highest replicative validity for document-based retrieval models. Despite this, we can conclude that field-based generation is more valid for use in semi-structured document retrieval experiments, since it was shown to have both higher predictive and replicative validity in overall.

Table 6.6. P-values of Kolmogorov-Smirnov test for different query generation methods on TREC queries. Queries generated with field-based method have higher p-values in overall.

Extent	P_{term}	DQL	PRM-S	PRM-D
Document	Uniform	0.003	0.000	0.041
	TF	0.090	0.000	0.005
	IDF	0.000	0.023	0.000
	TF*IDF	0.000	**0.160**	0.000
Field	Uniform	**0.085**	**0.323**	**0.276**
	TF	**0.105**	**0.667**	**0.570**
	IDF	**0.068**	0.013	0.008
	TF*IDF	**0.284**	0.021	0.022

6.2.2 Pseudo-desktop Collections

Since the validation methods described in Section 5.3.3 requires hand-written queries, we collected hand-written queries for the three pseudo-desktop collections by showing people a set of documents and asking for a query they would use to find each of the documents. Note that this procedure was later improved to create the DocTrack game described in Section 5.5.

Specifically, we first showed each participant a set of target documents. After a time period, we asked them to formulate a query based on their memory of a document assuming that the document is to be found in the desktop. Three people participated in this experiment and a total of 50 queries were manually generated for each email sub-collection of the three pseudo-desktops.

The result in Table 6.7 shows the same trends as the TREC collection, reconfirming the replicative validity of field-based generation methods, especially when query-terms were selected randomly or based on term frequency. Document-based generation methods show replicative validity only for some of the retrieval models. Since the sample size for the hand-written query set was smaller (50) than that of the TREC collection (150), we set a higher threshold (0.1) for the p-value.

Table 6.7. P-values of Kolmogorov-Smirnov test for different query generation methods in pseudo-desktop collections.

Extent	P_{term}	DQL	PRM-S	PRM-D
Document	Uniform	0.068	**0.417**	**0.129**
	TF	0.058	**0.619**	**0.244**
	IDF	0.000	**0.116**	0.003
	TF*IDF	0.000	**0.266**	0.007
Field	Uniform	**0.621**	**0.299**	**0.406**
	TF	**0.456**	**0.207**	**0.605**
	IDF	**0.110**	0.027	0.061
	TF*IDF	**0.227**	0.030	0.066

6.3 Field-based Search Models

In this section, we describe experiments for verifying the effectiveness of the field-based search models we proposed. We first describe experiments with the field-based retrieval methods in existing structured document collections, followed by experiments on the type prediction and final (merged) retrieval performance in simulated collections of personal information.

6.3.1 Evaluation in Structured Document Collections

Here we present experiments for evaluating field-based retrieval models in structured document collections. We introduce the collections and other experimental settings. We used three collections with different document and query characteristics, and varying numbers of relevant documents per query.

First, we used a well-known TREC collection for structured documents (emails). The TREC 2005 Enterprise track known-item task [23] used a crawl of the W3C mailing list, containing 198,394 documents with average length of 10kb. For each document, the indexed fields were *subject*, *body*, *to* (receiver), *date* and *from* (sender). Among the 150 queries provided, according to the TREC guideline, 25 were set aside for training of model parameters and the rest were used for testing. This collection has only one relevant document per query.

We also used the IMDB[2] collection, which consists of 437,281 documents. Each document here corresponds to a movie and was constructed from the text data downloaded from the IMDB website[3]. The fields were *title*, *year*, *releasedata*, *language*, *genre*, *country*, *location*, *colorinfo*, *cast*, *team*. We used 50 queries (10 for training and 40 for evaluation) developed in a previous study [51]. This collection has two relevant documents per query on average.

Finally, we used the Monster[4] job description collection composed of 1,034,795 documents. Here, the documents were longer, with mostly full-text content. Each document is composed of fields like *resumetitle*, *summary*, *jobtitle*, *school*, *experience*, *location* ,*skill* and *additionalinfo*. The 60 queries we used (20 for training and 40 for evaluation) were requests for job descriptions created by real users of the Monster service. This collection has 15 relevant documents per query on average.

During indexing, each word was stemmed using the Krovetz stemmer and standard stop words were eliminated. Indri[5] was used as a retrieval engine for all the retrieval experiments. Mean Average Precision (MAP) was used as the measure of retrieval performance for all experiments, since there were one or more relevant documents for each query with no grades in relevance judgments.

For baselines in our experiments, we used DQL [66], BM25F [70], MFLM [64]. We evaluate PRM-S and the Field Relevance Model against these baselines. Since we introduced several techniques for field relevance estimation, we evaluated the estimation based on the combination of sources (FRM) and the oracle estimation based on a set of known relevant documents (FRM$_O$).

[2]IMDB is the largest collection of movie resources on the web. http://www.imdb.com

[3]Available in http://www.imdb.com/interfaces#plain

[4]Monster is one of the largest websites for online hiring and job search. http://www.monster.com

[5]http://www.lemurproject.org

Table 6.8. The distribution of relevant fields in each collection. Each number denotes the number of query terms found in the respect field of the relevant documents.

Collection	Distribution of Relevant Fields
TREC	subject:307 body:220 to:44 date:11 from:1
IMDB	title:43 actors:30 genre:13 team:11 year:4 location:2 country:1
Monster	jobtitle:86 location:58 resumetitle:52 school:16 skill:19 experience:19 summary:1

Since each retrieval model required a different set of parameters to be tuned in advance, we used a training and test split for each query set. For parameters that required training for each document field, such as per-field weights w_j and b_j, we performed a coordinate ascent search using training queries which find the best-performing parameter combination. As for the field relevance model (FRM) we similarly found the mixture weights Λ that maximize the retrieval performance in the training queries.

6.3.1.1 Statistics of Field Relevance

We first show the distribution of relevant fields in each collection we used. For our measurement, we calculated the oracle field relevance estimate using relevant documents for each query, and then took the field with highest relevance for each query term. That is, these are the fields most likely to generate each query term from the field-level language model of relevant documents. Table 6.8 shows the statistics, where it is clear that the field relevance is spread across many fields in all the collections we tested. These results are consistent with the evidence in Section 3.2.2, where we presented the statistics of field distributions for queries from previous work.

Figure 6.1 shows the examples of Indri queries taken from each collection, where weights are again estimated using relevant documents for each query. We can see that each query term has matches in many different fields.

```
#combine(
  #wsum(1.000 Basic.(body))
  #wsum(1.000 Authentication.(body))
  #wsum(1.000 Microsoft.(subject))
  #wsum(0.925 IE.(subject) 0.075 IE.(body))
  #wsum(1.000 removed.(body)))
```

```
#combine (
  #wsum(1.000 tom.(actors))
  #wsum(1.000 hank.(actors))
  #wsum(0.819 1994.(year) 0.181 1994.(releasedate)))
```

```
#combine (
  #wsum(0.421 flight.(resumetitle) 0.364 flight.(jobtitle)
        0.185 flight.(experience) 0.029 flight.(school))
  #wsum(0.468 control.(resumetitle) 0.336 control.(jobtitle)
        0.169 control.(experience) 0.027 control.(school))
  #wsum(0.592 engineer.(jobtitle) 0.337 engineer.(resumetitle)
        0.060 engineer.(experience) 0.011 engineer.(school))
  #wsum(0.714 kansas.(location) 0.217 kansas.(school)
        0.068 kansas.(experience))
```

Figure 6.1. Indri query with oracle field relevance estimate from TREC, IMDB and Monster collection, respectively.

Table 6.9. Retrieval performance for three collections used. FRM_O is based on the oracle estimate of field relevance.

	DQL	BM25F	MFLM	PRM-S	FRM	FRM_O
TREC	0.542	0.597	0.601	0.624	0.668	0.794
IMDB	0.408	0.524	0.612	0.637	0.657	0.704
Monster	0.429	0.279	0.460	0.542	0.558	0.716

6.3.1.2 Retrieval Effectiveness

Table 6.9 shows retrieval results for all the collections. Although the difference varies for each collection, PRM-S and the variants of FRM consistently improves baseline methods in all collections we tested. Among baseline methods, field-based retrieval models (BM25F and MFLM) showed better performance except for the case of BM25F model in the Monster collection, which might be due to the parameter estimation using a small number (20) of training queries.

The improvements over DQL, BM25F and MFLM methods were statistically significant (using the paired t-test with p-value < 0.05) in all three collections we tested. Especially, the performance of FRM in TREC collection represents an improvement over an already strong baseline (the best performance among TREC submissions was 0.621 [22]).

Finally, *oracle* estimates of field relevance in FRM_O shows the upper bound of performance one can get with ideal field relevance estimation. Oracle estimates were derived from the known relevant documents in each test collection, as described in Section 3.2.4.2.

Note that the only difference between MFLM, PRM-S and FRM is how field weights are estimated. Since FRM employs the per-field weights in MFLM and the unigram collection field-language model in PRM-S as the sources of combination, we can infer that the additional sources used for FRM resulted in the improvement.

To gain further insights into the impact of different sources on retrieval effectiveness, we performed a feature ablation study where we omitted a set of sources from

Table 6.10. Feature ablation results in TREC collection. Each column denotes the performance where top-k document features (tug/tbg), bigram features (cbg/tbg), collection features (cbg/cug) and prior feature were omitted, respectively.

Features Omitted	None	tug/tbg	cbg/tbg	cbg/cug	prior
MAP	0.668	0.662	0.651	0.648	0.644

the estimation of field relevance. We denote here five sources used for field relevance estimation as *cug* (collection unigram field-level language model (FLM)), *cbg* (collection bigram FLM), *tug* (top-k documents unigram FLM), *tbg* (top-k documents bigram FLM), and *prior* (per-field weight estimated using training queries). More detailed descriptions can be found in Section 3.2.4.3.

The results in Table 6.10 shows the impact from the omission of each source group on performance. You can see that all the source groups have a positive impact on the performance, and the omission of *prior* has the most impact on performance. This shows the importance of having a reliable back-up method in case the per-term field relevance estimation is difficult.

6.3.1.3 Field Relevance Estimation and Retrieval Performance

Since we hypothesized that the performance advantage of FRM is based on improved estimation of field relevance, we then compared retrieval models in terms of the quality of field relevance estimates. For this experiment, we used three similarity metrics for field relevance introduced in Section 3.2.4.5, which measure the similarity of given estimates with the oracle estimates. The initial letter for each retrieval model (MFLM, PRM-S, FRM) is used as a subscript to denote the estimates used in different retrieval methods.

The results in Table 6.11 show that the field relevance model (FRM) improves estimation quality over MFLM and PRM-S, which use limited sources for field relevance estimation. This result is consistent with our expectation that per-field and

Table 6.11. Quality of estimated field relevance compared to oracle estimation using the aggregated per-term KL-divergence (KL), cosine similarity (Cos) and precision (P@1). Higher value means higher quality, except for KL.

	KL_M	KL_P	KL_F	Cos_M	Cos_P	Cos_F	$P@1_M$	$P@1_P$	$P@1_F$
TREC	2.994	1.099	0.821	0.636	0.719	0.765	0.528	0.582	0.642
IMDB	2.764	0.723	0.529	0.405	0.814	0.876	0.478	0.802	0.820
Monster	4.121	1.481	1.381	0.358	0.650	0.675	0.015	0.467	0.481

Table 6.12. The correlation of the difference in query-level retrieval performance and the quality of field relevance estimate between PRM-S and FRM.

	diff(MAP)	diff(Cos)	diff(KL)
diff(Cos)	0.448		
diff(KL)	-0.505	-0.851	
diff(Prec)	0.233	0.804	-0.563

per-term estimation in PRM-S is better than per-field estimates in MFLM, and that the quality of estimation is improved by the combination of sources for FRM.

We further quantified the impact of field relevance estimation quality on the performance in the TREC collection, measured in terms of the correlation in query-level differences. For each query, we took the difference in MAP and three metrics of quality for field relevance between PRM-S and FRM. We then calculated the correlation of the differences in the query-level retrieval performance and the quality of field relevance estimate.

Table 6.12 shows the results, where the difference in retrieval performance has high correlation with all three quality metrics of field relevance. Among the quality metrics for field relevance, Kullback-Leibler divergence and cosine similarity showed higher correlation to performance than precision. From this we can see that the improvement in FRM can be attributed to better estimation of field relevance.

We also provide an example with the query 'Basic Authentication Microsoft IE removed' that shows the difference in the field relevance estimate between PRM-S and FRM. Comparing with the oracle estimate in Figure 6.1, we can see that the

```
#combine(
  #wsum(0.530 Basic.(subject) 0.465 Basic.(body))
  #wsum(0.780 Authentication.(subject)
        0.215 Authentication.(body))
  #wsum(0.693 Microsoft.(from) 0.222 Microsoft.(to))
  #wsum(0.496 IE.(from) 0.196 IE.(to))
  #wsum(0.575 removed.(subject) 0.413 removed.(body)))
```

Figure 6.2. An Indri query example with field relevance estimate based on PRM-S.

```
#combine(
  #wsum(0.634 Basic.(subject) 0.364 Basic.(body))
  #wsum(0.762 Authentication.(body)
        0.238 Authentication.(subject))
  #wsum(0.400 Microsoft.(subject) 0.347 Microsoft.(from))
  #wsum(0.618 IE.(body) 0.21 IE.(subject))
  #wsum(0.709 removed.(body) 0.289 removed.(subject))
```

Figure 6.3. An Indri query example with field relevance estimate based on FRM.

field relevance estimate in PRM-S makes more mistakes for this query compared to that of FRM.

Most notably, the *from* field was given the highest weight for query term 'Microsoft' in PRM-S, whereas *subject* field correctly got the highest weight in FRM. Considering that the term is likely to appear in senders and receivers of an email very frequently, it is understandable that PRM-S incorrectly assigned it to those fields. However, the combination-based estimate in FRM helped avoid the same mistake.

6.3.2 Evaluation in Simulated Test Collections

In this section, we describe the experiments for verifying the effectiveness of the type prediction and retrieval methods in the simulated collections we created. Recall from Section 3.1 that the final retrieval score is the function of both type prediction score (collection-level) and retrieval score (document-level).

We used three pseudo-desktop collections with generated queries for the first experiment, where we compared several type prediction methods and showed the impact of type prediction on the final ranking. We then report on experiments using the CS collection where queries were collected by the DocTrack game.

Four retrieval methods were used for each sub-collection (DQL / PRM-S / PRM-D / Best) and four methods (Uniform / CQL / FQL / Oracle) were used for type prediction. Since we wanted to use only the retrieval methods for which we confirmed the validity of generated queries, we compared Document Query-Likelihood (DQL) [66], the Probabilistic Retrieval Model for Semi-structured data (PRM-S) and the interpolation between the two (PRM-D).

Among the type prediction methods, we compared only CQL and FQL for the pseudo-desktop experiment, since CQL was shown to be the most effective among collection scoring methods [83] and FQL is the extension of CQL for semi-structured document collections. Section 6.3.2.2 provides the comparison with other type prediction methods using the CS collection and queries from the DocTrack game.

With type-specific rank lists from sub-collection retrieval and collection scores from the type prediction component, we can produce the final rank list by rank-list merging algorithms. As introduced in Section 3.1, we use the well-known CORI algorithm for merging [14].

For the “Best” retrieval method, we used the retrieval method with the best aggregate performance for each sub-collection, making the assumption that the best-performing retrieval method is known in advance. For “Uniform” and “Oracle” collection scoring, we considered that each collection has the same chance of containing the relevant document (Uniform) or that we have the perfect knowledge of the collection that contains the relevant document (Oracle).

Table 6.13. Accuracy of type prediction in pseudo-desktop collections. The numbers indicate the percentage that the correct collection is ranked the highest for the corresponding type prediction method the collection.

	Jack	Tom	Kate
CQL	0.606	0.637	0.380
FQL	0.773	0.807	0.640

6.3.2.1 Pseudo-desktop Collections

Three pseudo-desktop collections described in Section 6.1.1 were used for these experiments. Each collection contained typical file types such as email, webpage and office documents related to each of three individuals. For each experiment, we generated 50 queries of average length 2 where target documents were taken from each sub-collection in proportion to the number of documents it contains. All the experiments were repeated three times since the query generation procedure involves randomness.

In Table 6.13, we compare the accuracy of type prediction in pseudo-desktop collections for the CQL and FQL methods, where FQL shows a clear improvement over the CQL method. Although this result should be interpreted with some reservations because we are using simulated queries, the same trend was found in the experiment using manual queries. We also observe that both methods show reasonable performance in the *Jack* and *Tom* collections, which contain far more email documents than other types. From this, we can conclude that both methods are relatively robust against an imbalance of sub-collection sizes.

We report the retrieval performance for the same queries in Table 6.14. The first noticeable trend is that both the choice of type-specific retrieval model and type prediction method has a big impact on the final result. In particular, Oracle type prediction was much better than the FQL method, which in turns outperformed CQL across all collections. On the other hand, the Best retrieval method was not much better than the PRM-D and PRM-S methods.

Table 6.14. Retrieval performance in three pseudo-desktop collections using different type-specific retrieval methods and type prediction methods. The numbers indicate the MAP for the final merged ranking.

	Jack				Tom				Kate			
	Uniform	CQL	FQL	Oracle	Uniform	CQL	FQL	Oracle	Uniform	CQL	FQL	Oracle
DLM	0.129	0.159	0.270	0.331	0.104	0.123	0.192	0.224	0.126	0.12	0.237	0.294
PRM-S	0.152	0.212	0.326	0.403	0.150	0.209	0.289	0.348	0.232	0.239	0.383	0.532
PRM-D	0.148	0.219	0.335	0.403	0.155	0.204	0.289	0.346	0.250	0.245	0.387	0.538
Best	0.154	0.225	0.336	0.414	0.157	0.217	0.302	0.361	0.241	0.245	0.388	0.542
Average	0.146	0.204	0.317	0.388	0.141	0.188	0.268	0.320	0.212	0.212	0.349	0.477

6.3.2.2 CS Collection

We report on experiments using the computer science (CS) collection, where the documents of various types are collected from many public sources in the UMass Computer Science department. More details of the collection can be found in Section 6.1.2.

We used 528 queries to obtain query-log feature (QQL) values and training parameters for other features, since some of features required data for estimation. The rest (456) of the queries were used to evaluate the type prediction performance of features and combination methods by 10-fold cross-validation. For the retrieval experiments, since many queries did not return any documents, we used only queries where the relevant document was ranked in the Top 50 result set during the DocTrack game.

Table 6.15 summarizes the prediction accuracy results, comparing two of the best-performing single feature runs (CQL / FQL) and combination methods (Grid / RankSVM / MultiSVM). Recall that the combination methods employ scores from several type prediction methods as features, including CQL and FQL methods. Again, the numbers indicate the percentage that the correct collection is ranked the highest for the corresponding type prediction method.

The result shows that all the combination runs improved performance over the best single feature runs given by FQL, which outperformed CQL in this collection as well. MultiSVM was shown to be the most effective among the combination methods.

Table 6.15. Accuracy of type prediction for best-performing individual type prediction methods and combination methods in the CS collection.

Method	CQL	FQL	Grid	RankSVM	MultiSVM
Accuracy	0.708	**0.743**	0.747	0.758	**0.808**

Table 6.16. Significance test results for type prediction accuracy in a CS collection. Each cell shows the p-value of paired t-test between the accuracy of two methods.

Method	CQL	FQL	Grid	RankSVM	MultiSVM
CQL		0.03	0.00	0.00	0.00
FQL			0.69	0.27	0.02
Grid				0.41	0.02
RankSVM					0.07

This is understandable considering that we had one target collection for each query, which is a natural setting for multi-class classification. RankSVM was slightly better than Grid but the difference was not significant.

The result of a significance test is reported in Table 6.16, which shows that the performance differences between CQL and all the other methods are significant with a p-value of 0.05 and that MultiSVM outperforms all the other methods significantly with a p-value of 0.1 (using paired t-test). Overall, this means that the suggested type prediction method (FQL) improves the performance of the CQL method and that the combination of features improves the performance further.

Table 6.17 shows the retrieval performance, comparing four retrieval methods (DQL / PRM-S / PRM-D / Best) and the same set of type prediction methods as above in addition to Oracle and Uniform methods.

Table 6.17. Retrieval performance in a CS collection using different type-specific retrieval methods and type prediction methods.

	Uniform	CQL	FQL	Grid	RankSVM	MultiSVM	Oracle	Average
DQL	0.343	0.507	0.53	0.552	0.563	0.556	0.674	0.526
PRM-S	0.349	0.501	0.518	0.518	0.551	0.547	0.674	0.520
PRM-D	0.360	0.518	0.536	0.536	0.567	0.564	0.694	0.537
Best	0.372	0.548	0.564	0.590	0.596	0.594	0.720	0.563
Average	0.356	0.518	0.537	0.549	0.569	0.565	0.691	

The result mostly shows the same trends as the pseudo-desktop collections despite a big difference in experimental conditions (recall that queries were algorithmically generated for the pseudo-desktop collections). FQL was better than CQL and all the combination methods outperformed CQL and FQL significantly (with paired t-test using the p-value of 0.05).

The only exception was that the performance of MultiSVM was slightly worse than RankSVM. Given the superior prediction accuracy of MultiSVM, it seems that the procedure of converting the SVM output into the type prediction score caused some problems. We can also see that Oracle type prediction method and the Best retrieval method outperform other methods, which leaves room for further improvement in both type-specific retrieval and type prediction.

6.4 Associative Browsing Model

In this section, we present the experimental results for the associative browsing model. Based on the simulation results as well as the log data collected during the DocTrack game, we first analyze the role of associative browsing for the known-item finding task. We then focus on the quality of the suggestions generated by the learning method described in Section 4.2.

6.4.1 Evaluation based on DocTrack User Studies

We first evaluate the effectiveness of associative browsing in the known-item finding task. We performed two rounds of game-style user studies in which participants were asked to find randomly chosen documents. We use the term ‘session’ to denote the process of finding each target document. We have 290 sessions from Round 1 and 142 sessions from Round 2, where users had the option to make use of the concept space in Round 2.

Table 6.18. Statistics of the sessions with search and browsing. Users had the option to user the concept space in Round 2.

Round	Total	Browsing used	Successful
1st	290	42 (14%)	15 (36%)
2nd	142	43 (30%)	32 (74%)

6.4.1.1 Usage and Effectiveness of Browsing

As we can see from Table 6.18, the percentage of sessions during which users chose to browse as well as search was 14% (or 42 sessions) for Round 1 and 30% (or 43 sessions) for Round 2. The significant percentage of users who decided to use the browsing indicates that users would like to have the option of browsing in addition to search. Moreover, the fact that we have a higher browsing usage rate in the second round seems to suggest that the concept space provided further motivations for browsing.

Out of the 42 sessions in Round 1 involving both searching and browsing, 36% (or 15 sessions) of them were successful, i.e., the user found the required document. For Round 2, this percentage is 74% (or 32 sessions). The higher successful rate in the second round can be attributed to the presence of the concept layer. In fact, many users commented that they could find the target document using the concepts as an intermediate step. Another interesting comment is that browsing was helpful for thinking of good query words.

6.4.1.2 Efficiency of Browsing

To investigate the efficiency of user's individual actions, we then looked at the average time for each type of user action. The results in Table 6.19 shows that it takes a much shorter time for the user to make a click decision on the ranked list of concepts compared to the ranked list of documents. This is understandable considering that the titles of concepts (e.g., person names) are much easier to read

Table 6.19. Average time for each type of user's action.

User's Action	Average Time (seconds)
Click on a Concept	8.50
Click on a Document	13.63
Keyword Search	15.14

than the titles of typical documents. By comparison, keyword search took longer than both types of browsing action.

In summary, the analysis of user behavior shows that the users find associative browsing helpful for known-item finding in some cases, and the use of concept layer makes the interaction more effective and efficient.

6.4.2 Evaluation based on Simulated User Model

We now report on the experiments using the simulated user model described in Section 5.4, comparing the outcome with those from the user study. As for the parameters of keyword search model, we used the language model of a document for P_{term}, and set the query length to 1.5 on average, following the average query length from previous studies [30] [27]. For the associative browsing component of the user model, based on the user model described in Section 5.4.2, we experimented with three levels of user's knowledge (*random*, *informed* and *oracle*), fan-out of 1 to 3, and breadth-first search (BFS) and depth-first search (DFS) browsing strategies.

In modeling the transitions between states, we made several assumptions. First, the user chooses to browse only when the ranked list returned by keyword search looks marginally relevant. Second, there is no transition from browsing back to search. It is left for future work to relax these assumptions to create a more realistic model of the transitions.

We use the term *marginally relevant* for the case where the target document is located between the rank position of 11 to 50. Whenever the target document is found within top 10 positions, we consider the session as *successful* and the interaction is finished. The session is unsuccessful if the user fails to find the target document within 10 trials.

For each target item, we ran the simulated user model described above. In order to keep the quality of ranking consistent, we used a simple vector space model based on the Lucene search engine toolkit[6] for both keyword search and associative browsing (i.e., we used only the content similarity feature among the features introduced in [50]). Finally, since the simulation involves the random generation of user behavior, we ran all the experiments 10 times and report the average results.

6.4.2.1 Usage and Effectiveness of Browsing

We now discuss the results from the simulation experiments. Table 6.21 shows the success ratio of browsing aggregated across three models of user's knowledge and fan-outs. As mentioned above, the success ratio here denotes the portion of sessions where browsing led to *success* among all sessions whose initial queries were *marginally relevant*.

In aggregate, the results in the first row of Table 6.20 show that browsing was used for about 15% of sessions. Since we designed the user model so that it starts browsing only when initial search result is marginally relevant, this indicates the quality of keyword query we used — 15% of queries found the target document between the rank of 11 to 50. If we look at the success ratio results, about 42% of sessions using the browsing were successful in the end, showing that associative browsing effectively complements keyword search.

We also compare the results with those from our previous user-based evaluation in Table 6.20. We found that the success ratio of the simulation study is similar to the ratio of successful browsing sessions based on the user study we reported in

[6]http://lucene.apache.org

Table 6.20. The ratio of the sessions where the simulated user model chose to use browsing and the choice of browsing led to a successful retrieval, in comparison to the user study results.

Evaluation type	Total	Browsing used	Successful
Simulation	63,260	9,410 (14.8%)	3,957 (42.0%)
User Study	290	42 (14.5%)	15 (35.7%)

our previous work, which suggests that the assumptions we made in the simulation experiments are reasonable.

6.4.2.2 Varying Simulation Parameters

With respect to different levels of user knowledge, even when the user browses randomly without any knowledge of the target document, the chance for success is almost 30%, showing that associative browsing is an effective alternative to search even when we do not make any assumptions about user knowledge. At the same time, however, the success ratio of browsing was no higher than 50%, showing that there are cases where the target document simply cannot be reached by browsing.

An interesting trend in Table 6.21 is the relationship between the user's level of knowledge and fan-out. We originally expected that the *oracle* user would outperform others at all fan-outs, yet it was found that higher fan-out (more exploration) only hurts the *oracle* user whose level of knowledge is very high yet always make a locally optimal decision.

In contrast, the success ratio of the *random* user increased with higher fan-out, which shows that exploration is valuable only when the user's level of knowledge is low. Overall, the *informed* user with a fan-out of two gave the best performance.

Table 6.21. Success ratio of browsing for marginally relevant queries.

	random	informed	oracle
FO1	0.337	0.401	**0.424**
FO2	0.408	**0.453**	0.441
FO3	**0.442**	0.436	0.426

Table 6.22. Average length of successful browsing session.

	random	informed	oracle
FO1	1.417	1.391	**1.266**
FO2-BFS	2.186	1.904	**1.661**
FO3-BFS	2.083	1.959	**1.814**
FO1	1.417	1.391	**1.266**
FO2-DFS	2.280	1.928	**1.257**
FO3-DFS	2.327	1.805	**1.323**

As for the fan-out, we can see that more fan-out leads to more success, as we might expect. What is surprising is that higher fan-out did not help the *oracle* user model as it did the *random* or *informed* user model. The success rate for the *oracle* user is much higher than others when fan-out is 1, yet at fan-out 3, the *informed* user shows higher success rate than the *oracle* user. Although this may seem counterintuitive at first glance, presumably some level of exploration might be required in getting to the target document.

We then looked at the efficiency of using browsing for known-item finding, which we measured by the average length of successful browsing sessions — how many clicks it took for the user to find the target document. Here we compared three levels of user's knowledge and two browsing strategies — BFS and DFS for each fan-out.

The results in Table 6.22 show that one or two clicks are usually sufficient to get to the target document by browsing. Comparing different levels of user knowledge, the *oracle* user model is always more efficient, followed by *informed* and *random*. We can conclude that the user's level of knowledge has a direct influence on the efficiency of browsing.

Among different browsing behaviors, it is clear that higher fan-out (more exploration) leads to lower efficiency in most cases, as we would expect. A less obvious trend is that the *random* user is more efficient with BFS strategy (exploration first), while the *oracle* user is more efficient with DFS strategy (exploitation first). Again, we can infer that higher levels of knowledge makes exploitation more valuable than exploration.

In summary, the simulation experiments show that associative browsing provides an effective (30-40% of success) and efficient (within 1–2 clicks) way of getting to the target document when keyword search is marginally relevant. Comparison of results across different levels of user knowledge and browsing behavior reveals the influence of various aspects of the user on the effectiveness and efficiency of associative browsing for known-item finding.

6.4.3 Evaluation of Browsing Suggestions

For evaluating the effectiveness of suggestions for browsing, we employed three test collections. Two volunteers, Person 1 and Person 2, used some of their personal information to create two of the collections. The former contains 8,841 documents and 368 concepts and the latter contains 9,441 documents and 945 concepts. Both collections are mostly composed of emails, webpages, and desktop files. As far as the clicks we used for training are concerned, Person 1 clicked 145 times on the ranked list of concepts and 58 times on the ranked list of documents. Person 2 had 196 clicks and 204 clicks on concepts and documents, respectively.

The third dataset is the CS collection introduced in Section 6.1.2. This collection is composed of 7,984 documents and 650 concepts. For click data, we used the data from a user with highest number of clicks (CS/Top1). We also experimented with the aggregate data from the five users with most clicks (CS/Top5). The number of items and clicks are summarized in Table 6.23.

For learning methods, we used our own implementation of Iterative Grid Search and SVM^{rank} [40], which is a popular implementation of RankSVM. To facilitate the training of SVM^{rank}, each feature value was scaled to values that were approximately

Table 6.23. Number of documents, concepts, and clicks in the case of document similarity and concept similarity experiments for each of the collections we used.

	#Items		#Clicks	
	Document	Concept	Document	Concept
Person 1	8841	368	58	129
Person 2	9411	945	204	196
CS/Top1	7984	650	145	42
CS/Top5	”	”	309	220

Table 6.24. Concept ranking performance (MRR) for the single-feature and combination methods. 10-fold cross-validation was used for grid search and RankSVM (SVM).

Collection	title	content	tag	time	string	cooc	occur	Grid	SVM
Person 1	0.097	0.229	0.194	0.136	0.136	**0.241**	0.151	0.236	**0.277**
Person 2	0.037	0.350	0.403	0.221	0.310	**0.516**	0.234	**0.581**	0.509
CS/Top1	0.142	0.179	**0.289**	0.235	0.107	0.191	0.195	0.255	**0.433**
CS/Top5	0.184	0.127	0.170	0.155	0.100	0.158	**0.222**	0.301	**0.340**

between 0 and 1. We also used 10-fold cross validation for training feature weights and evaluating the system.

In order to measure retrieval performance, we used the Mean Reciprocal Rank (MRR), which is the average of reciprocal of click positions. We also used clicks as relevance judgments, which is an approximate yet reasonable assumption made in many studies.

6.4.3.1 Quality of Browsing Suggestions

Here we present the evaluation results on the quality of ranking for browsing suggestions. We compared the performance obtained when each feature was used by itself and when three combination methods were used—feature values with equal weights (*Uniform*), weights obtained with grid search (*Grid*) and with RankSVM (*SVM*), respectively. Note that *title* and *content* are term vector similarity features, where the title and the content field were used for constructing term vectors, respectively.

Table 6.24 shows the concept ranking results for each feature and combination method. Regarding the single-feature results, different features turned out to be useful

Table 6.25. Document ranking performance (MRR) for the single-feature and combination methods. 10-fold cross-validation was used for grid search and RankSVM (SVM).

Collection	title	content	tag	time	topic	path	type	concept	Grid	SVM
Person 1	0.392	**0.480**	0.063	0.296	0.229	0.274	0.183	0.264	**0.500**	0.494
Person 2	0.334	**0.564**	0.268	0.372	0.184	0.137	0.092	0.187	**0.592**	0.478
CS/Top1	0.074	0.097	0.065	0.114	**0.140**	0.098	0.070	**0.140**	**0.156**	0.098
CS/Top5	0.081	0.138	0.05	0.114	**0.151**	0.132	0.062	0.129	**0.150**	0.133

for each collection. Specifically, we found that *co-occurrence* is the most effective feature in Person 1's and Person 2's collection, while *occurrence* and *tag* was the best in CS/Top5 and CS/Top1, respectively. From this we can conclude that there are considerable variations in the value of each feature depending on the collection and the click behavior.

Among the combination methods, RankSVM performed the best for all collections except for Person 2's, where Grid Search performed the best. We also observe that, although different features perform the best for each collection, combination methods are consistently better than single-feature methods. From these results, we can draw the conclusion that feature combination is beneficial for finding concept similarity.

As far as the document ranking task is concerned, Table 6.25 shows a slightly different trend. Term vector similarity using the *content* field is far more important than any other features in the case of Person 1 and Person 2. This makes intuitive sense because documents typically contain more textual content. This results in more accurate term vectors and subsequently better term vector similarity estimates. The best feature for the CS collection was topic similarity.

In the case of combination methods, grid search performed better than any feature used by itself, while RankSVM was not as effective as it was in concept ranking. Although the performance margin between combination and single-features methods is small, given that it is hard to know which feature would work the best a priori, we can conclude that the feature combination should be used for generating document suggestions as well.

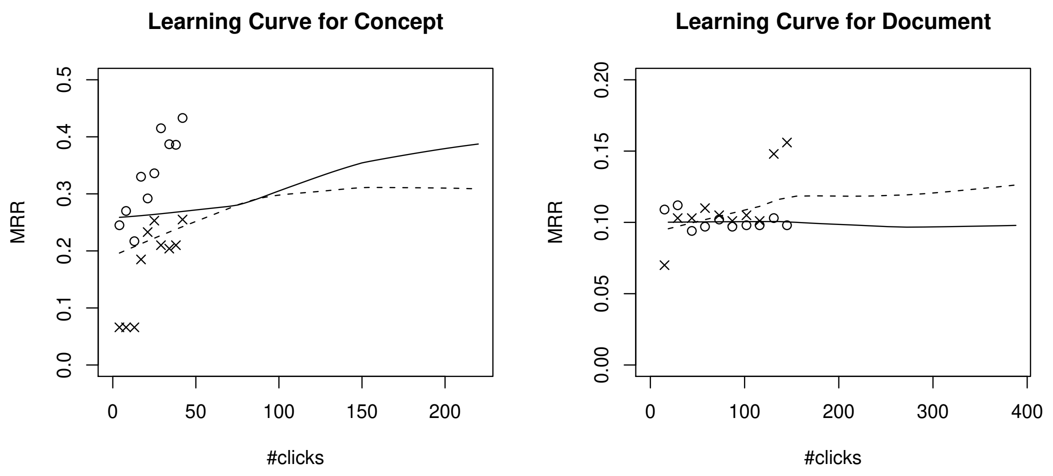


Figure 6.4. Learning curve for concept and document ranking on CS/Top1 collection. The circles and crosses denote the ranking quality of RankSVM and grid search (respectively) when trained using the user's own click data, whereas solid and dashed lines represent the performance of RankSVM and grid search (respectively) when trained on the click data of the five users.

6.4.3.2 Analysis of the Performance

To analyze the performance of learning methods using different sets of click data, we compared the learning curve for the CS/Top1 collection trained using the user's own click data versus the aggregate click data from five users. Figure 6.4 shows the result we obtained. Circles and crosses represent the learning curve based on the user's own clicks, whereas solid and dashed lines represent the learning curve based on aggregated clicks of the five users.

We can see that training with the user's own click data results in better performance, even if we have much smaller number of clicks. From this result, we can conclude that different users have different preferences, which the suggested learning method effectively adapts to.

Another trend is that RankSVM shows better average performance and a steeper learning curve in the case of the concept ranking task, whereas the opposite is observed in the document ranking task. We hypothesize that the diversity among top-ranked documents led to the generation of noisy pairwise constraints from the click data. Upon further inspection, we found that the average number of skipped items was higher in the document similarity experiment compared to the concept similarity experiment.

6.5 Summary

In this chapter, we presented the experimental results for the methods we proposed in previous chapters. We first described the test collections we built, and evaluated the validity of queries we generated. We found that field-based query generation improves document-based query generation in terms of both predictive and replicative validity.

We then presented experimental results for the retrieval methods we proposed earlier—field-based search models and associative browsing model. For field-based search models, our evaluation on several structured document collections showed that the field-based retrieval methods (PRM-S and FRM) were significantly more effective than the baselines. In analyzing the results, we found that the quality of the field relevance estimation is largely responsible for the improvement.

Using the simulated test collections (pseudo-desktop and CS), we demonstrated that improving the type prediction method can produce significantly better final retrieval performance. We also found that the suggested type prediction methods show performance superior to competitive baselines in both collections we tested. Finally, our results show that the combination-based type prediction method can improve type prediction performance compared to existing methods.

Our evaluation for the associative browsing model based on both a user study and simulation suggests that the model is useful for the known-item finding task, especially when the concept space is used for browsing in addition to the document space. We showed that the effectiveness of each association type varies according to the collection and the user. Furthermore, our experimental results in three collections show that the combination of features significantly improves the quality of suggestions for browsing.

CHAPTER 7
CONCLUSIONS

In this chapter, we conclude this thesis. We first summarize the main results from the thesis, and then conclude the thesis by discussing its implications in a broad context. Finally, we discuss the limitations of current work and present possible extensions for the future.

7.1 Summary

This thesis propose a general retrieval and evaluation framework for personal information retrieval (PIR), with the goal of building a foundation for future research in the area. The techniques introduced in this thesis also improve the state-of-the-art in the relevant areas of research. We briefly summarize the main contributions and experimental results from this thesis.

For the field-based search method, we showed the value of exploiting field-level evidence for both retrieval and type-prediction methods. Motivated from the observation that users associate each part of the query with different structural elements, we put forth the idea of field relevance and related estimation techniques by extending the notion of relevance for structured document retrieval.

Table 7.1. Retrieval performance for three collections used. FRM_O is based on the oracle estimate of field relevance using relevant documents.

	DQL	BM25F	MFLM	PRM-S	FRM	FRM_O
TREC	0.542	0.597	0.601	0.624	0.668	0.794
IMDB	0.408	0.524	0.612	0.637	0.657	0.704
Monster	0.429	0.279	0.460	0.542	0.558	0.716

Table 7.2. Accuracy of type prediction for best-performing individual type prediction methods and combination methods in the CS collection.

Method	CQL	FQL	Grid	RankSVM	MultiSVM
Accuracy	0.708	**0.743**	0.747	0.758	**0.808**

Table 7.3. The ratio of the sessions where the simulated user model chose to use browsing and the choice of browsing led to a successful retrieval, in comparison to the user study results.

Evaluation type	Total	Browsing used	Successful
Simulation	63,260	9,410 (14.8%)	3,957 (42.0%)
User Study	290	42 (14.5%)	15 (35.7%)

Experiments in several standard test collections show that the retrieval methods based on field relevance improve retrieval performance over strong baselines (Table 7.1). We also show how field relevance can be estimated based on known relevant document (the rightmost column of Table 7.1). We introduced a type prediction method combining field-level evidences, as well as a type prediction method based on the combination of evidence (Table 7.2).

For the associative browsing model, we introduced a technique for automatically building associations between documents and concepts, where the association strengths are calculated by combining features. We showed how associative browsing can complement keyword search based on both user studies and simulation experiments (Table 7.3). Finally, we demonstrated that the proposed learning framework for generating browsing suggestions can improve the quality of suggestions based on a small amount of click feedback from the user (Tables 7.4 and 7.5).

To address the evaluation challenge, we introduced a set of simulation techniques by which we can create all the components necessary for PIR evaluation. The tech-

Table 7.4. Concept ranking performance (MRR) for the single-feature and combination methods in the CS collection.

Collection	title	content	tag	time	string	cooc	occur	Grid	SVM
CS/Top1	0.142	0.179	**0.289**	0.235	0.107	0.191	0.195	0.255	**0.433**
CS/Top5	0.184	0.127	0.170	0.155	0.100	0.158	**0.222**	0.301	**0.340**

Table 7.5. Document ranking performance (MRR) for the single-feature and combination methods in the CS collection.

Collection	title	content	tag	time	topic	path	type	concept	Grid	SVM
CS/Top1	0.074	0.097	0.065	0.114	**0.140**	0.098	0.070	**0.140**	**0.156**	0.098
CS/Top5	0.081	0.138	0.05	0.114	**0.151**	0.132	0.062	0.129	**0.150**	0.133

niques exploit the task (known-item finding) and the collection (structured documents) characteristics of PIR. We also proposed a three-stage evaluation framework where simulated evaluation techniques can be sensibly combined with the traditional user study methods.

7.2 Conclusions

Ever since the idea of the personal computer was popularized, personal information management has been one of major functions of a computing system. As we have more and more personal information in electronic form, the problem of retrieving personal information has become increasingly important. The landscape of personal information management is still rapidly evolving with the changing paradigms of computing and the introduction of new technologies, which only increases the challenges.

Starting from the nature of personal information and how users access their own information, this thesis presented a combination of retrieval and evaluation techniques upon which future research can build. Actually, it was these characteristics of the domain that enabled the development of many techniques introduced here. Here we revisit the characteristics of PIR, and how they relate to this thesis.

The existence of multiple collections and document structure in the form of type-specific metadata became the foundation for the design of field-based retrieval model and type prediction methods. The observations that users tend to orienteer when they find information in their own collection motivated the development of the associative browsing model. Focusing on the most common task in PIR (known-item finding) allowed us to build evaluation techniques by simulating the query generation procedure and designing a game-based user study.

While this focus on PIR contributes the future development of the domain, many of the proposed techniques have application well beyond the area of PIR. Structured data and documents are becoming increasingly common, yet it is unlikely that such structure would be found in users' queries, since users would not have the ability or incentive to write queries with complex structure. Therefore, the general idea of mapping the user's query into different structural elements of the collection would be applicable in various areas of IR.

Specifically, while we evaluated the field-based retrieval models in flat fielded documents, the proposed notion of field relevance can be applicable to other scenarios. For instance, in XML retrieval where elements form a hierarchical relationship, we can define the mapping between the user's query terms and structural elements of each collection document. The problem of keyword search over databases also deals with relations composed of multiple attributes, each of which can be a target of a user's query terms.

The combination of multiple information access methods is also a common situation found in many domains. For instance, e-commerce websites such as Amazon.com provide search, faceted filtering and associative browsing for its users. However, the evaluation of IR in such an information seeking environment where multiple methods are combined remains a challenge. In this regard, the evaluation framework described here where we dealt with the combination of search and browsing based on simulated user models and game-based studies can be illuminating for future investigation.

Finally, while we considered the privacy of data as a concern for PIM research, in many other areas of research, getting the collection and user interaction data is a significant challenge. Building simulated test collections, as we showed in the pseudo-desktop and CS collections, can be a solution here. Many of the proposed techniques for test collection generation and user study would be applicable to other areas of research.

7.3 Future Work

As many of proposed techniques represent the first step in a new direction, this thesis has several limitations.

First, while focusing on the development of a general retrieval model, this thesis leave out many details of implementation. We did not deal with the retrieval features of any particular document type, as well as the considerations of nor did we fully consider efficiency and user interface issues.

From the evaluation perspective, we avoided instrumentation-based user studies that require the implementation of systems, by focusing on various forms of simulated evaluation. While we showed the validity of our evaluation based on some statistical techniques and by direct comparison of simulation results with corresponding user study results, some aspects of PIR would still require the evaluation based on actual users and their tasks. In what follows, we describe possible extensions of the proposed techniques.

- **Improving the Field Relevance Model:** We proposed field-based retrieval models with improved performance, and our analysis showed that the quality of per-term field weight estimation was a main source of the improvement. However, our results also showed that there still exists a wide gap between the current performance and what is possible, which leaves opportunities for future research. Also, the relationship of field weighting and the underlying retrieval model should be investigated for the proposed field weighting scheme to be applied to other retrieval frameworks.

- **Fluid Combination of Search and Browsing within a Session** The proposed retrieval model assumes a separation between search and browsing. In other words, at any given point, the retrieval is based on either keyword query or an item. However, we can envision a hybrid interaction model where search and browsing are combined in a single session. As an example, the user can type in a search keyword and choose a document in the same page, which functions as a context for retrieving more documents.

- **Incorporating Context for Generating Suggestions:** Given that browsing naturally leads to several sequences of interaction, one possibility of future work is to improve the suggested browsing model by incorporating the user's session context. The proposed model is stateless in that it makes suggestions only based on what the user is currently looking at. By exploiting user session contexts available in the form of browsing history, we believe that the quality of suggestions can be improved, given that a single document or a concept may not be discriminative enough to generate effective results.

- **Cognitive Model of Query Generation:** Existing query generation models make many assumptions about the user's query generation behavior, some of which do not necessarily correspond to what a human user might do. As a solution, we can build a more elaborate model of a user's memory, and then develop a query generation procedure based on the model. Such model would be capable of simulating many features of human memory, such as forgetting and confusion, which are important aspects of query formulation.

- **Naturalistic User Study:** We evaluated the proposed retrieval model based on several variants of simulation methods. While the evaluation results demonstrate the effectiveness of the proposed methods, more conclusive claim may be made from a naturalistic user study where a PIR system is deployed to users' machines and used for an extended period. Such an evaluation will reveal the characteristics of the retrieval methods in the context of actual user's information and tasks.

BIBLIOGRAPHY

[1] Adar, Eytan, Teevan, Jaime, and Dumais, Susan T. Large scale analysis of web revisitation patterns. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 1197–1206.

[2] Agrawal, Sanjay, Chaudhuri, Surajit, and Das, Gautam. Dbxplorer: enabling keyword search over relational databases. In *SIGMOD Conference* (2002), p. 627.

[3] Allan, James, Leuski, Anton, Swan, Russell, and Byrd, Donald. Evaluating combinations of ranked lists and visualizations of inter-document similarity, 2001.

[4] Amer-Yahia, Sihem, and Lalmas, Mounia. XML search: languages, INEX and scoring. *SIGMOD Record 35*, 4 (2006), 16–23.

[5] Arguello, Jaime, Callan, Jamie, and Diaz, Fernando. Classification-based resource selection. In *CIKM '09* (New York, NY, USA, 2009), ACM, pp. 1277–1286.

[6] Arguello, Jaime, Diaz, Fernando, Callan, Jamie, and Crespo, Jean-Francois. Sources of evidence for vertical selection. In *SIGIR '09* (New York, NY, USA, 2009), ACM, pp. 315–322.

[7] Azzopardi, Leif, de Rijke, Maarten, and Balog, Krisztian. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR '07* (New York, NY, USA, 2007), ACM, pp. 455–462.

[8] Bao, Shenghua, Duan, Huizhong, Zhou, Qi, Xiong, Miao, Cao, Yunbo, and Yu, Yong. Research on expert search at enterprise track of trec 2006. In *15th Text REtrieval Conf.* (2006).

[9] Bendersky, Michael, Metzler, Donald, and Croft, W. Bruce. Learning concept importance using a weighted dependence model. WSDM '10, ACM, pp. 31–40.

[10] Bergman, Ofer, Beyth-Marom, Ruth, Nachmias, Rafi, Gradovitch, Noa, and Whittaker, Steve. Improved search engines and navigation preference in personal information management. *ACM Trans. Inf. Syst. 26*, 4 (2008), 1–24.

[11] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent dirichlet allocation. *J. Mach. Learn. Res. 3* (2003), 993–1022.

[12] Bollegala, Danushka, Matsuo, Yutaka, and Ishizuka, Mitsuru. Measuring semantic similarity between words using web search engines. In *WWW '07* (New York, NY, USA, 2007), ACM, pp. 757–766.

[13] Calado, Pável, da Silva, Altigran S., Vieira, Rodrigo C., Laender, Alberto H. F., and Ribeiro-Neto, Berthier A. Searching web databases by structuring keyword-based queries. In *CIKM '02* (New York, NY, USA, 2002), ACM, pp. 26–33.

[14] Callan, James P., Lu, Zhihong, and Croft, W. Bruce. Searching distributed collections with inference networks. In *SIGIR '95* (New York, NY, USA, 1995), ACM, pp. 21–28.

[15] Chau, Duen Horng, Myers, Brad, and Faulring, Andrew. What to do when search fails: finding information by association. In *CHI '08* (New York, NY, USA, 2008), ACM, pp. 999–1008.

[16] Chen, Jidong, Guo, Hang, Wu, Wentao, and Wang, Wei. imecho: an associative memory based desktop search system. In *CIKM '09* (New York, NY, USA, 2009), ACM, pp. 731–740.

[17] Chernov, Sergey, Demartini, Gianluca, Herder, Eelco, Kopycki, Micha, and Nejdl, Wolfgang. Evaluating personal information management using an activity logs enriched desktop dataset. In *Personal Information Management Workshop at CHI 2008* (April 2008).

[18] Chernov, Sergey, Serdyukov, Pavel, Chirita, Paul-Alexandru, Demartini, Gianluca, and Nejdl, Wolfgang. Building a desktop search test-bed. In *ECIR* (2007), pp. 686–690.

[19] Cimino, JJ, Elkin, PL, and Barnett, GO. As we may think: the concept space and medical hypertext. *Information Processing and Management 29*, 3 (1993), 313–324.

[20] Coffman, Joel, and Weaver, Alfred C. A framework for evaluating database keyword search strategies. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (New York, NY, USA, 2010), CIKM '10, ACM, pp. 729–738.

[21] Cohen, Sara, Domshlak, Carmel, and Zwerdling, Naama. On ranking techniques for desktop search. *ACM Trans. Inf. Syst. 26*, 2 (2008), 1–24.

[22] Craswell, Nick, and Hugo Zaragoza, Stephen Robertson. Microsoft cambridge at trec-14: Enterprise track. In *The Fourteenth Text REtrieval Conference* (2005).

[23] Craswell, Nick, and Vries, Arjen P. De. Overview of the trec-2005 enterprise track. In *In The Fourteenth Text REtrieval Conf. Proc.* (2005).

[24] Croft, W. B., and Thompson, R. H. I^3R : A new approach to the design of document retrieval system. Tech. rep., Amherst, MA, USA, 1987.

[25] Croft, W. Bruce, and Turtle, Howard R. Retrieval strategies for hypertext. *Information Processing and Management 29*, 3 (1993), 313–324.

[26] Cronen-Townsend, Steve, Zhou, Yun, and Croft, W. Bruce. Predicting query performance. In *SIGIR '02* (New York, NY, USA, 2002), ACM, pp. 299–306.

[27] Cutrell, Edward, Robbins, Daniel, Dumais, Susan, and Sarin, Raman. Fast, flexible filtering with Phlat. In *CHI '06: Proceedings of the SIGCHI conference* (New York, NY, USA, 2006), ACM, pp. 261–270.

[28] Davis, G., and Thomson, D. Memory in context: Context in memory. Wiley.

[29] Dong, Xin, and Halevy, Alon Y. A platform for personal information management and integration. In *CIDR* (2005), pp. 119–130.

[30] Dumais, Susan, Cutrell, Edward, Cadiz, JJ, Jancke, Gavin, Sarin, Raman, and Robbins, Daniel C. Stuff I've seen: a system for personal information retrieval and re-use. In *SIGIR '03* (New York, NY, USA, 2003), ACM, pp. 72–79.

[31] Elsweiler, David, Baillie, Mark, and Ruthven, Ian. What makes re-finding information difficult? a study of email re-finding. In *ECIR* (2011), pp. 568–579.

[32] Elsweiler, David, Harvey, Morgan, and Hacker, Martin. Understanding re-finding behavior in naturalistic email interaction logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 35–44.

[33] Elsweiler, David, Losada, David E., Toucedo, José C., and Fernandez, Ronald T. Seeding simulated queries with user-study data for personal search evaluation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 25–34.

[34] Elsweiler, David, and Ruthven, Ian. Towards task-based personal information management evaluations. In *SIGIR '07* (New York, NY, USA, 2007), ACM, pp. 23–30.

[35] Greenberg, Saul, and Buxton, Bill. Usability evaluation considered harmful (some of the time). In *Proceeding of the SIGCHI conference* (New York, NY, USA, 2008), CHI '08, ACM, pp. 111–120.

[36] Guo, Lin, Shao, Feng, Botev, Chavdar, and Shanmugasundaram, Jayavel. Xrank: Ranked keyword search over xml documents. In *SIGMOD Conference* (2003), pp. 16–27.

[37] Harvey, Morgan, and Elsweiler, David. Exploring query patterns in email search. In *Proceedings of the 34th European conference on Advances in Information Retrieval* (Berlin, Heidelberg, 2012), ECIR'12, Springer-Verlag, pp. 25–36.

[38] Hauff, Claudia, and Houben, Geert-Jan. Cognitive processes in query generation. In *ICTIR* (2011), pp. 176–187.

[39] Hristidis, Vagelis, and Papakonstantinou, Yannis. Discover: Keyword search in relational databases. In *VLDB* (2002), pp. 670–681.

[40] Joachims, Thorsten. Optimizing search engines using clickthrough data. In *KDD '02* (New York, NY, USA, 2002), ACM, pp. 133–142.

[41] Jones, William. *The Future of Personal Information Management*. Synthesis Lectures on Information Concepts,Retrieval, and Services. Morgan & Claypool Press, 2012.

[42] Jones, William, and Teevan, Jaime. *Personal Information Management*. University of Washington Press, 2008.

[43] Kaplan, Craig, Fenwick, Justine, and Chen, James. Adaptive hypertext navigation based on user goals and context. *User Modeling and User-Adapted Interaction 3*, 3 (1993), 193–220.

[44] Karger, David R., Bakshi, Karun, Huynh, David, Quan, Dennis, and Sinha, Vineet. Haystack: A general-purpose information management tool for end users based on semistructured data. In *CIDR* (2005), pp. 13–26.

[45] Kim, Jin Young, Bakalov, Anton, Smith, David A., and Croft, W. Bruce. Building a semantic representation for personal information. In *In Proceedings of CIKM'2010, Toronto, Ontario, Canada* (2010).

[46] Kim, Jin Young, and Croft, W. Bruce. Retrieval experiments using pseudo-desktop collections. In *in Proceedings of CIKM'2009, Hong Kong, China* (2009), pp. 1297–1306.

[47] Kim, Jin Young, and Croft, W. Bruce. Ranking using multiple document types in desktop search. In *In Proceedings of SIGIR '10* (New York, NY, USA, 2010), ACM, pp. 50–57.

[48] Kim, Jin Young, and Croft, W. Bruce. A field relevance model for structured document retrieval. In Proceedings of ECIR '12: 34th European Conference on Information Retrieval.

[49] Kim, Jin Young, Croft, W. Bruce, and Smith, David A. Evaluating associative browsing model for by simulations. In *In Proceedings of HCIR'2011 Workshop, Mountain View, CA, USA* (2011).

[50] Kim, Jin Young, Croft, W. Bruce, Smith, David A., and Bakalov, Anton. Evaluating associative browsing model for personal information. In *In Proceedings of CIKM'2011, Glasgow, Scotland, UK* (2011).

[51] Kim, Jin Young, Xue, Xiaobing, and Croft, W. Bruce. A probabilistic retrieval model for semi-structured data. In *In Proceedings of ECIR '09* (2009), Springer, pp. 228–239.

[52] Lavrenko, Victor. *A generative theory of relevance.* PhD thesis, 2004. AAI3152722.

[53] Lavrenko, Victor, and Croft, W. Bruce. Relevance based language models. SIGIR '01, ACM, pp. 120–127.

[54] Lavrenko, Victor, Yi, Xing, and Allan, James. Information retrieval on empty fields. In *HLT-NAACL* (2007), pp. 89–96.

[55] Lee, Chia-Jung, Croft, W. Bruce, and Kim, Jin Young. Evaluating search in personal social media collections. In Proceedings of WSDM '12, 5th ACM International Conference on Web Search and Data Mining.

[56] Leuski, Anton, and Allan, James. Interactive information retrieval using clustering and spatial proximity. *User Modeling and User-Adapted Interaction 14*, 2-3 (June 2004), 259–288.

[57] Lin, Jimmy, and Smucker, Mark D. How do users find things with pubmed?: towards automatic utility evaluation with user simulations. In *SIGIR '08* (New York, NY, USA, 2008), SIGIR '08, ACM, pp. 19–26.

[58] Lu, Chang-Tien, Shukla, Manu, Subramanya, Siri H., and Wu, Yamin. Performance evaluation of desktop search engines. In *IRI* (2007), pp. 110–115.

[59] Lucarella, Dario. A model for hypertext-based information retrieval. 81–94.

[60] M. Géry, C. Largeron, F. Thollard. Probabilistic document model integrating xml structure. *Proceedings in INEX* (2007), 139–149.

[61] Ma, Hao, Chandrasekar, Raman, Quirk, Chris, and Gupta, Abhishek. Improving search engines using human computation games. In *CIKM* (2009), pp. 275–284.

[62] Metzler, Donald, and Bruce Croft, W. Linear feature-based models for information retrieval. *Information Retrieval 10* (June 2007), 257–274.

[63] Metzler, Donald, and Croft, W. Bruce. Combining the language model and inference network approaches to retrieval. *IPM 40*, 5 (2003), 735–750.

[64] Ogilvie, Paul, and Callan, Jamie. Combining document representations for known-item search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference* (New York, NY, USA, 2003), ACM, pp. 143–150.

[65] Petkova, Desislava, Croft, W. Bruce, and Diao, Yanlei. Refining keyword queries for xml retrieval by combining content and structure. ECIR '09, Springer-Verlag, pp. 662–669.

[66] Ponte, Jay, and Croft, W. Bruce. A language modeling approach to information retrieval. ACM, ACM, pp. 275–281.

[67] Press, William, Teukolsky, Saul, Vetterling, William, and Flannery, Brian. *Numerical Recipes in C*, 2nd ed. Cambridge University Press, Cambridge, UK, 1992.

[68] Ravasio, Pamela, Schär, Sissel Guttormsen, and Krueger, Helmut. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact. 11*, 2 (June 2004), 156–180.

[69] Robertson, S. E., and Jones, K. Sparck. Relevance weighting of search terms. *Journal of the American Society for Information Science 27*, 3 (1976), 129–146.

[70] Robertson, Stephen, Zaragoza, Hugo, and Taylor, Michael. Simple BM25 extension to multiple weighted fields. In *In Proceedings of CIKM '04* (New York, NY, USA, 2004), ACM, pp. 42–49.

[71] Ruthven, Ian. Re-examining the potential effectiveness of interactive query expansion. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (New York, NY, USA, 2003), SIGIR '03, ACM, pp. 213–220.

[72] Sauermann, Leo, and Heim, Dominik. Evaluating long-term use of the gnowsis semantic desktop for pim. In *ISWC '08* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 467–482.

[73] Seo, Jangwon, and Croft, W. Bruce. Blog site search using resource selection. In *CIKM '08* (New York, NY, USA, 2008), ACM, pp. 1053–1062.

[74] Shah, Sam, Soules, Craig A. N., Ganger, Gregory R., and Noble, Brian D. Using provenance to aid in personal file search. In *ATC'07: 2007 USENIX* (Berkeley, CA, USA, 2007), USENIX Association, pp. 1–14.

[75] Shokouhi, Milad, and Si, Luo. Federated search. *Found. Trends Inf. Retr. 5*, 1 (Jan. 2011), 1–102.

[76] Si, Luo, and Callan, Jamie. Relevant document distribution estimation method for resource selection. In *SIGIR '03* (New York, NY, USA, 2003), ACM, pp. 298–305.

[77] Si, Luo, Jin, Rong, Callan, Jamie, and Ogilvie, Paul. A language modeling framework for resource selection and results merging. In *CIKM '02* (New York, NY, USA, 2002), ACM, pp. 391–397.

[78] Smucker, Mark D., and Allan, James. Find-similar: similarity browsing as a search tool. In *SIGIR '06* (New York, NY, USA, 2006), ACM, pp. 461–468.

[79] Teevan, Jaime, Adar, Eytan, Jones, Rosie, and Potts, Michael A. S. Information re-retrieval: repeat queries in yahoo's logs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 151–158.

[80] Teevan, Jaime, Alvarado, Christine, Ackerman, Mark S., and Karger, David R. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04* (New York, NY, USA, 2004), ACM, pp. 415–422.

[81] Thomas, Paul. Server characterisation and selection for personal metasearch.

[82] Thomas, Paul, and Hawking, David. Evaluation by comparing result sets in context. In *Proceedings of the 15th ACM international conference on Information and knowledge management* (New York, NY, USA, 2006), CIKM '06, ACM, pp. 94–101.

[83] Thomas, Paul, and Hawking, David. Server selection methods in personal metasearch: a comparative empirical study. *Inf. Retr. 12*, 5 (2009), 581–604.

[84] Tulving, E., and Thomson, D. Encoding specificity and retrieval processes in episodic memory. In *Psychological Review* (England, 1973), pp. 352–373.

[85] Tyler, Sarah K., and Teevan, Jaime. Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining* (New York, NY, USA, 2010), WSDM '10, ACM, pp. 191–200.

[86] von Ahn, Luis, and Dabbish, Laura. Designing games with a purpose. *Commun. ACM 51*, 8 (2008), 58–67.

[87] White, Ryen W., and Roth, Resa A. *Exploratory Search: Beyond the Query-Response Paradigm.* Morgan & Claypool, 2009.

[88] White, Ryen W., Ruthven, Ian, Jose, Joemon M., and Rijsbergen, C. J. Van. Evaluating implicit feedback models using searcher simulations. *ACM Trans. Inf. Syst. 23*, 3 (July 2005), 325–361.

[89] Yahyaei, Sirvan, and Monz, Christof. Applying maximum entropy to known-item email retrieval. In *ECIR* (2008), pp. 406–413.

[90] Yi, Xing, Allan, James, and Croft, W. Bruce. Matching resumes and jobs based on relevance models. In *SIGIR* (2007), pp. 809–810.

[91] Zhao, Le, and Callan, Jamie. A generative retrieval model for structured documents. In *In Proceedings of CIKM '08* (New York, NY, USA, 2008), ACM, pp. 1163–1172.