

**RETRIEVAL AND EVALUATION TECHNIQUES
FOR PERSONAL INFORMATION**

A Dissertation Outline 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

March 2011

Department of Computer Science

RETRIEVAL AND EVALUATION TECHNIQUES FOR PERSONAL INFORMATION

A Dissertation Outline Presented

by

JINYOUNG KIM

Approved as to style and content by:

W. Bruce Croft, Chair

James Allan, Member

David A. Smith, Member

Michael L. Lavine, Member

Andrew G. Barto, Department Chair
Department of Computer Science

ABSTRACT

RETRIEVAL AND EVALUATION TECHNIQUES FOR PERSONAL INFORMATION

MARCH 2011

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.

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.

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

	Page
ABSTRACT	iii
 CHAPTER	
1. INTRODUCTION	1
1.1 Personal Information Retrieval	1
1.1.1 A Typical Scenario	1
1.1.2 Problem Characteristics	2
1.2 Our Approach	3
1.2.1 Retrieval Models	3
1.2.2 Evaluation Methods	4
1.3 Contributions	5
1.3.1 Completed Work	5
1.3.2 Proposed Work	6
1.4 Organization	6
 2. RELATED WORK	 7
2.1 Desktop Search	7
2.2 Semi-structured Document Retrieval	7
2.3 Associative Browsing	8
2.4 Other Areas of Related Work	9
2.4.1 Known-item Search	9
2.4.2 Federated Search	9
2.4.3 Human Computation Game	9
 3. TERM-BASED SEARCH MODEL	 10
3.1 Type-specific Retrieval Method	10
3.1.1 Document Query-Likelihood	11
3.1.2 Probabilistic Retrieval Model for Semi-structured Data	11
3.1.3 The Mixture of PRM-S and Document Query Likelihood	12
3.2 Type Prediction Method	12
3.2.1 Using Document Metadata Fields for Type Prediction	12
3.2.2 Combining Type Prediction Methods	13
3.3 Summary & Proposed Work	13

4. ASSOCIATIVE BROWSING MODEL	14
4.1 Data Model	14
4.2 Creating Suggestions for Browsing	14
4.2.1 Features	15
4.2.2 Learning Feature Weights	17
4.3 Summary & Proposed Work	17
5. SIMULATION-BASED EVALUATION METHOD	18
5.1 Evaluating Term-based Search by Simulation	18
5.1.1 Collecting Documents	18
5.1.2 Generating Known-Item Queries	19
5.1.2.1 Field-Based Query Generation	19
5.1.3 Evaluating Equivalence to Manual Queries	20
5.1.3.1 Verifying Predictive Validity	21
5.1.3.2 Verifying Replicative Validity	21
5.2 Pseudo-desktop Collections	21
5.2.1 Generated Queries	22
5.3 Summary & Proposed Work	23
6. GAME-BASED EVALUATION METHOD	24
6.1 DocTrack Game	24
6.1.1 CS Collection	25
6.1.2 DocTrack Search Game	25
6.1.3 DocTrack Search & Browsing Game	26
6.2 Summary & Proposed Work	27
7. EVALUATION RESULTS	28
7.1 Evaluation of Term-based Search Model	28
7.1.1 Type-specific Retrieval Performance	28
7.1.2 Type Prediction and Merged Retrieval Performance	29
7.1.2.1 Pseudo-desktop Collections	29
7.1.2.2 CS Collection	30
7.2 Evaluation of Associative Browsing Model	31
7.2.1 The Role of Associative Browsing	32
7.2.2 Retrieval Performance	32
7.3 Summary & Proposed Work	34
BIBLIOGRAPHY	35

CHAPTER 1

INTRODUCTION

The focus of this thesis is designing and evaluating techniques for personal information retrieval (PIR). We define PIR as the process involved in a person accessing their own information stored in digital form. 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. While web search has changed how people access information on the web, 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 in many different ways, 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. We also introduce evaluation methods 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.

In the remainder of this chapter, we first give an overview of the problem domain by describing the characteristics of PIR. Based on this analysis, we then introduce the approaches taken by this thesis in detail, followed by the research contributions to be made.

1.1 Personal Information Retrieval

1.1.1 A Typical Scenario

Before we characterize the problem of personal information retrieval (PIR), 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

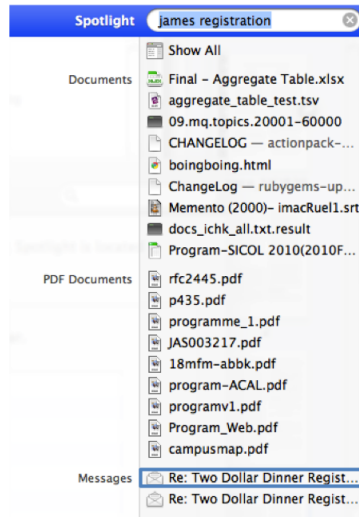


Figure 1.1. A typical scenario of personal information retrieval. The target document is highlighted with a box.

email, she types in a query (‘james registration’) on her computer. She gets the following results in Figure 1.1, which contains the email she had in mind around the end of the ranked list.

Although there would be many other scenarios of PIR, the case above exemplifies the most common scenario: the search for a known item on the user’s desktop. In this thesis, we focus on this case, but with a approach that generalizes beyond that.

1.1.2 Problem Characteristics

The proper characterization of a given problem is important for developing an effective solution in general, and is particularly so for the study of PIR which has different meanings for different people. Here we intend to define the problem clearly by describing several characteristics 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 documents of many 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 user, it has been suggested that people mostly find known-items in their own document collections. This is understandable in that almost any item in personal information is created and accessed by users, and that web search is what people usually turn to for finding information about a new topic. We primarily focus on the known-item search task for this thesis.

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 informa-

tion needs of many individuals. This long-term interaction provides opportunities for the system to better understand and serve the user, and 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 party 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.

1.2 Our Approach

1.2.1 Retrieval Models

Based on the characteristics described above, we propose using term-based search and associative browsing as the primary retrieval models for personal information. More specifically, term-based search provides a capability for a user to search for documents across all one’s digital collections using a single query. Associative browsing enables the user to browse through documents by following the chain of associations.

These 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 file organization, the tags of blog posts). Also, these two methods are complementary and can be naturally combined in a single system. For instance, users can rely on associative browsing in case the term-based search is unsuccessful.

Here, we provide a concrete example on 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 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 1.2, 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).

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

Secondly, we intend to build adaptive methods which can use the natural interactions with the user to improve effectiveness without conscious effort. Our term-based search model uses query logs to improve the performance of type prediction. The associative browsing model exploits click feedback to refine suggestions for browsing.

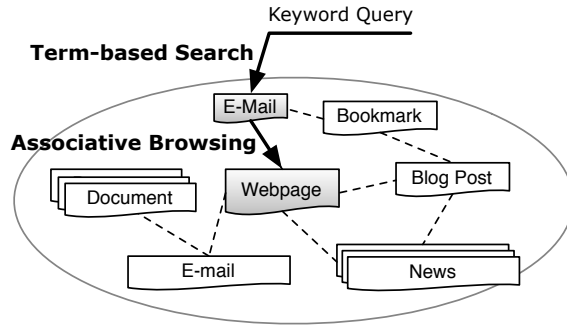


Figure 1.2. An illustration of how term-based search and associative browsing can be combined. Dotted lines represent the associations between documents and concepts. Directed lines denote how a user can access the target webpage by using term-based search first, and then associative browsing.

1.2.2 Evaluation Methods

Previous studies of PIR often involved a diary study—deploying the system in a real environment and having it evaluated by actual users. Although this kind of evaluation method has its own benefits, it requires considerable resources. Moreover, the collections and usage logs from these studies are not open to other researchers because they include private information. In evaluating proposed retrieval methods, we propose a set of techniques that address this privacy issue by simulating some components of evaluation.

First of all, we propose a simulation-based evaluation method which does not require any human involvement by simulating user’s interaction with the system. 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 game-based user study where we motivate participants to contribute the log data. 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.

Figure 1.3 shows three components of PIR evaluation and how suggested evaluation methods replaced each component of diary study with simulated components. The game-based evaluation method (DocTrack) is based on employing simulated collections and tasks, while the simulation-based evaluation method (Pseudo-desktop) uses algorithmically generated user interaction logs as a substitute for human behavior. Combining both the simulation and game-based evaluation methods, we can accomplish the goal of realistic and low-cost evaluation.

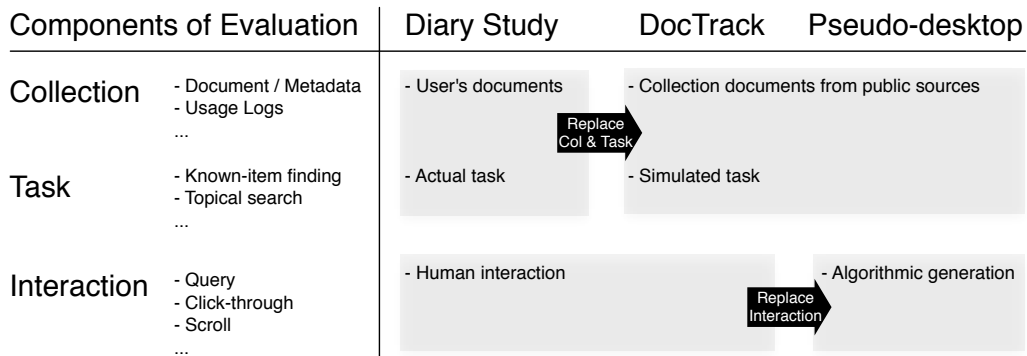


Figure 1.3. Components of PIR evaluation and the comparison of three evaluation methods.

1.3 Contributions

1.3.1 Completed Work

The initial work on this thesis has made the following contributions, which resulted in five publications (three papers and two posters). In what follows, we listed major contributions along with corresponding references.

- A novel term-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 method 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. [31]
- 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 and the CS collection. [30]
- 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. [30]
- An adaptive method for suggesting associations between documents or terms 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. [28]
- A method called Pseudo-desktop for building simulated test collections for evaluating personal information retrieval. Specific contributions within the pseudo-

desktop method include novel methods for generating and evaluating queries for known-item search. [29]

- 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 in this environment. [30]

1.3.2 Proposed Work

For the work that needs to be done, we plan to make the following research contributions within the time frame specified. We introduce details of each proposed activity at the end of the corresponding chapter.

- **2011/3 – 2011/5** We will improve the PRM-S further, based on more accurate prediction of the mapping between query-terms and document fields. We will also work on improving the query-generation method. (Chapter 3 and 5)
- **2011/9 – 2011/12** We will develop a method for building simulated user interaction logs for evaluating both term-based search and associative browsing in a unified manner. (Chapter 5)
- **2011/9 – 2011/12** We will verify the output of the game-based evaluation method using the data collected from more realistic environments. (Chapter 6)
- **2012/1 – 2012/5** We will run additional evaluations in other collections and discuss findings in the final thesis. (Chapter 7)

1.4 Organization

This proposal is organized as follows: in Chapter 2, we briefly overview related work in the areas of structured document retrieval, personal information management and desktop search, and others. In Chapter 3 and 4, we describe the term-based search model and associative browsing model in detail, respectively.

In Chapter 5 and 6, we change our focus to the evaluation techniques and describe simulation-based evaluation method and game-based evaluation method in detail, respectively. Finally, in Chapter 7, 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 of this proposal, since it relies on the understanding of proposed evaluation methods as well as retrieval methods.

CHAPTER 2

RELATED WORK

2.1 Desktop Search

Desktop search systems such as Stuff I've Seen [23] and Phlat [20] showed that user interaction is a significant issue in desktop environment, and that the *date* is 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 [40] are useful features. Learning feature weights with training data [14] has also been found to be effective in the desktop environment.

Thomas et al. [45] regarded desktop search as a meta-search problem where the results from many servers are merged. They compared several server selection methods using documents collected from various sources, concluding that a selection method based on Kullback-Leibler divergence [42] 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.

The evaluation of desktop search or, in general, personal information retrieval (PIR), has been considered a challenging problem [26] because real desktop collections are not available for research due to privacy concerns. The performance evaluation of major commercial desktop search engines was tried [32] in standard IR evaluation settings, using TREC Robust track data. Chernov et al. [13] [12] proposed a method for creating a testbed for desktop search by collecting documents and queries collaboratively, yet no experimental validation was done. Elswailer [24] 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.

2.2 Semi-structured Document Retrieval

As 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 [37]. This straightforward combination of field-level scores was found to have limitations, resulting in efforts such as BM25F [39]. Recently, an adaptation of score combination and smoothing method was suggested [49] for the language modeling approach to IR, based on the search engine Indri [36] which supports combining evidence from multiple fields.

The INEX initiative is a major study of XML retrieval [1]. 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 [34] 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.

Other recent work [38] 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 [8] describe a method of ranking candidate structured queries that is similar to the PRM-S retrieval model described in Section 3.1.2, although it was applied and evaluated differently.

2.3 Associative Browsing

Since the early days of IR, researchers have been interested in the combination of search and browsing for accessing document collections. Lucarella [33] described a system with a network of concepts and documents which provides search and browsing capability in a complementary manner. *I³R* system developed by Croft and Thompson [17] also assumes scenario where documents returned by a user's initial query provide a starting point for subsequent browsing. Another paper by Croft and Turtle [18] 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.

Kaplan et al. [27] described a navigation scheme that adapts to user behavior. Smucker and Allan[43] 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.

Associative browsing models for personal information were introduced in previous studies [11] [10] [22]. 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 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. [7] measured the semantic similarity based on the results from a web search engine. Smucker et al.[43] used the unigram language model of a given document as a query to find similar documents. The suggested model for associative browsing is novel in 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.

2.4 Other Areas of Related Work

2.4.1 Known-item Search

The TREC 2005 Enterprise Track [16] provided a known-item email retrieval task, where a set of emails and corresponding queries were given. Among the participants, the BM25F model [15] combined a variety of document fields and other features such as the year and the thread structure to get good effectiveness. Another approach [48] combined different independent sources to improve the performance of known-item search. PRM-S retrieval model in Section 3.1.2 outperformed the methods described above in a recent evaluation [29] of ours. For the evaluation of known-item search, Azzopardi et al.[4] suggested query generation method, which is adapted in Pseudo-desktop method described in Section 5.1.

2.4.2 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 [9] and KL-Divergence [42] treat collections as large documents and apply document scoring techniques for scoring collections. Other methods, such as ReDDE [41] model the distribution of relevant documents for each collection.

Recently, Arguello et al. proposed a classification approach [2] [3] 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 3.2 is similar to this work but we use features and evaluation methods more suitable for our problem domain.

2.4.3 Human Computation Game

Human computation games [47] 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. [35] 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 Chapter 6 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

TERM-BASED SEARCH MODEL

In this chapter we propose a model for term-based search. In our model, 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. We first explain methods we used for the retrieval of sub-collections corresponding to each file type, focusing a novel retrieval method for semi-structured document retrieval. Then we introduce two novel type prediction methods—field-based query-likelihood and adaptive method. For merging type-specific results and type prediction scores into a single result, we used the well-known CORI algorithm [9].

The following notation will be used throughout this chapter. We assume that a query $Q = (q_1, \dots, q_m)$ is composed of m words and each collection C contains documents with n field types (F_1, \dots, F_n) where n can be different for each collection. Each document d in the collection includes fields (f_1, \dots, f_n) , where each field is marked using lowercase letters to distinguish it from the corresponding field type in the collection.

3.1 Type-specific Retrieval Method

The first step in our retrieval model is ranking documents from each sub-collection. Although any ranking method appropriate for each collection can be used, we employ retrieval models typically used for semi-structured document retrieval, which includes document query-likelihood (DQL), the probabilistic retrieval model for semistructured data (PRM-S) and the interpolation of the DQL and PRM-S (PRM-D). We

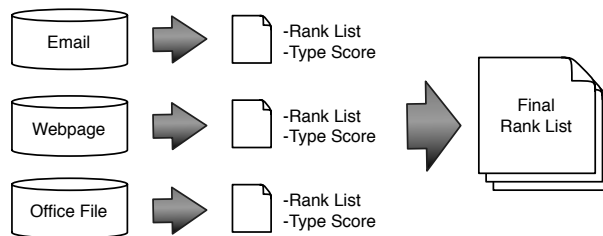


Figure 3.1. Suggested retrieval model for desktop search.

explain these models in the following sections. Among these, we focus on the PRM-S method, which is a novel retrieval model we proposed in [31] for semi-structured document retrieval.

3.1.1 Document Query-Likelihood

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

$$P(Q|d) = \prod_{i=1}^m (P_{QL}(q_i|d)) \quad (3.1)$$

Although DQL method does not take the structure of documents into account, we can use DQL model for our situation by ignoring the field structure and treating the whole document as a bag of words. Our previous work [29] showed that DQL can outperform other methods for some of document types.

3.1.2 Probabilistic Retrieval Model for Semi-structured Data

The probabilistic retrieval model for semistructured data (PRM-S) [31] scores documents by combining field-level query-likelihood scores similarly to other field-based retrieval models [37]. The main feature of the PRM-S model is that weights for combining field-level scores are estimated based on the predicted mapping between query terms and document fields, which can be efficiently computed based on 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 $P_M(w|F_j)$.

$$P_M(F_j|w) = \frac{P_M(w|F_j)P_M(F_j)}{\sum_{F_k \in F} P_M(w|F_k)P_M(F_k)} \quad (3.2)$$

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 $P_{QL}(w|f_j)$ into a document score, as follows:

$$P(Q|d) = \prod_{i=1}^m \sum_{j=1}^n P_M(F_j|q_i)P_{QL}(q_i|f_j) \quad (3.3)$$

This model was shown to have better performance than other field-based retrieval methods, such as the mixture of field language models [37] and BM25F [39], for a

semi-structured document retrieval task using the IMDB [31] and TREC email [29] collections.

3.1.3 The 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 user 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 3.4, thereby striking a balance between these two.

$$P(Q|d) = \prod_{i=1}^m ((1 - \lambda) \sum_{j=1}^n P_M(F_j|q_i) P_{QL}(q_i|f_j) + \lambda P_{QL}(q_i|d)) \quad (3.4)$$

where λ is the parameter that controls the interpolation ratio.

3.2 Type Prediction Method

In this section, we introduce our type prediction methods in detail. We introduce a new type prediction method that exploits document metadata. Then, we explain our framework for combining type prediction methods using several learning methods.

3.2.1 Using Document Metadata Fields for Type Prediction

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 [31], 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 [42] 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 3.1.2, 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, \dots, F_n) , we can combine the language model score of each field as follows:

$$FQL(Q, C) = \prod_{q \in Q} comb_{F_i \in C}(P(q|F_i)) \quad (3.5)$$

Here, F_i is a smoothed language model of the i th 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.

3.2.2 Combining Type Prediction Methods

In addition to the FQL method introduced above, there are many metrics which can be used for type prediction [2] [3]. 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.

Among type-prediction methods, we used the query-likelihood of collection language model (CQL), the query-likelihood of query logs (QQL), the geometric average of top document scores (GAVG), ReDDE [41], Query Clarity [19] and dictionary-base matching. More details can be found in [30]. We also employed three learning methods with different objective functions: grid search of parameter values, a multi-class classifier and a rank-learning method. More details of our learning framework can be found in [30].

3.3 Summary & Proposed Work

In this chapter, we suggested a term-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 type-specific retrieval method, we introduced a novel retrieval method for structured documents called Probabilistic Retrieval Model for Semistructured data (PRM-S). For 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.

For the completion of this thesis, we plan to improve the components of suggested term-based retrieval model further. Currently, the PRM-S method estimates the mapping between query-terms and document fields independently. Since there exist dependencies between query-terms and between terms in different parts of a document, incorporating these dependencies can potentially improve the quality of mapping probability estimation.

For instance, let us assume that ‘Meg Ryan’ is an actress who mostly featured in ‘romance’ films, and we need to estimate field mappings for the query ‘meg ryan romance’. If we estimated that the query word ‘meg’ can be mapped into *cast* field, it is likely that it may have a word ‘romance’ in *genre* field, because documents with ‘meg’ in *cast* field is likely to have ‘romance’ in *genre* field. In other words, the dependency between terms across different fields may provide further evidence in the inference of the mapping between query words and document fields. And the estimation for this dependency can be done using collection term statistics, as it was the case for the estimation of mapping probability.

CHAPTER 4

ASSOCIATIVE BROWSING MODEL

Associative browsing denotes the process of going through personal information by following a chain of associations. It has several benefits for personal information retrieval. First of all, studies in cognitive psychology [21] [46] show that people remember facts primarily by associations, which explains the intuitive appeal of associative browsing. Also, Teevan et al. [44] suggest that many people tend to find information by a series of small steps (orienteering) instead of using keyword search.

In this chapter, we first introduce an associative browsing model for personal information. We then describe our method for creating associations between items in detail.

4.1 Data Model

On a high level, our associative browsing model is composed of information *items* and the *associations* between them, as we can see from Figure 4.1. Items are defined as information objects with textual contents. These objects can be the *documents* collected from many sources (e.g., desktop files, emails, calendar items), or the *concepts* (e.g., person names, events, etc.).

In our model, concepts denote entities and terms of interest to the user that can be extracted from document metadata (e.g., sender and receiver of a email). They are similar to labels or facets in that they provide an abstract layer of organizing documents, yet they are distinctive in that they form a space of association on their own.

Since items can have a text representation—title, URI, content and metadata—the user can perform a keyword search for any of them. When the user clicks on one of search results, the system suggests related documents and concepts. As seen from Figure 4.1, in a scenario where a user’s initial keyword search returns only a marginally relevant concept (person), the user can browse into the target document (webpage) through another document (email) associated with both the concept and the target document.

4.2 Creating Suggestions for Browsing

A major challenge in implementing associative browsing is creating associations between items, because their quality is of great importance to user’s satisfaction.

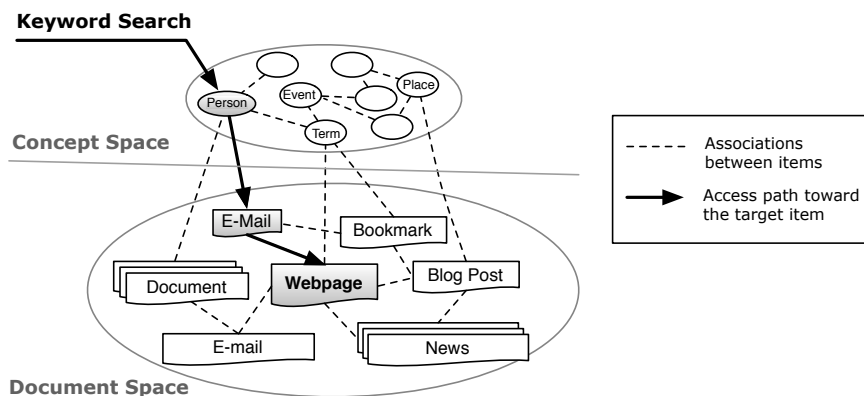


Figure 4.1. An illustration of the suggested associative browsing model. Dotted lines represent the association between documents and concepts. Directed lines denote how a user can access the target webpage by using keyword search and associative browsing.

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.

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. As a solution, 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 the ranked list of related concepts or documents, generated by combining many measures of association with appropriate weights.

Since we suggest the weighted combination of similarity measures for the ranking, another important task is finding appropriate weights for each feature. We address this issue by using users' 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 capture the similarity between concepts and between documents. Note that some of features are applicable only for the calculation of concept or document similarity. If that is the case, then the text in brackets after the name of the feature will reflect that.

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.

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.

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.

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. We get a high value if two strings have many similar substrings. For example, two concepts like ‘Database’ and ‘Data Model’ would be considered similar.

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.

Occurrence (concept) This feature counts the number of times a concept has occurred in the document collection. This metric is intended to capture the popularity of each concept. Similarly to the co-occurrence, it can be used only for finding concept similarity.

Topical Similarity (document) This feature relies on the topic model Latent Dirichlet Allocation (LDA) [6]. 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.

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, normalized by the number of words. Also, since each document has a type (e.g., email, pdf, etc.), whether two documents are of the same type can be another feature.

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.

4.2.2 Learning Feature Weights

In order to learn feature weights, we used two algorithms with different characteristics—iterative grid search and RankSVM [25]. In terms of the training goal, grid search finds the set of parameters that maximizes the target metric, whereas the goal of RankSVM is to predict the pairwise preference relation with the highest accuracy. Also, while grid search uses each click as a relevance judgment, RankSVM interprets each click as a pairwise preference.

4.3 Summary & Proposed Work

In this chapter, we proposed an associative browsing model of personal information. Our associative browsing model is composed of items (concepts and documents) that can be tagged 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 with appropriate weights learned using click feedback from the user.

For the completion of this thesis, we propose improving suggested browsing model by incorporating user’s session context. Suggested model is stateless in that it makes suggestions only based on what the user is currently looking at. By exploiting user’s session contexts available in the form of browsing history, we believe that the quality of suggestions can be improved.

For instance, knowing that a user is looking at the concept of ‘research scientist’ does not very informative if the user has many concepts which can be associated with ‘research scientist’, yet knowing that the concept the user previous visited is ‘information retrieval’ helps us to suggest research scientists in the area of information retrieval with higher ranks than the scientists from other areas of research.

CHAPTER 5

SIMULATION-BASED EVALUATION METHOD

In this chapter, we describe an evaluation method for personal information retrieval (PIR) based on simulation. Simulation in this context refers to automatically generating a part of test collections for evaluating IR systems.

Simulation-based evaluation is valuable for the evaluation of personal information retrieval for several reasons: first of all, we can get preliminary evaluation results for retrieval models before we perform an expensive user study. Secondly, we can experiment with a variety of assumptions on user, which is an important aspect of evaluating a PIR method.

However, given the nature of PIR which aims at satisfying user's information needs in the end, it would be impossible for simulation techniques to completely replace the study with actual user involvement.

In what follows, we first describe our method for building a simulated collection including queries and relevant documents, and collections based on the method. As a future work, we propose a novel simulation-based method for evaluating the scenario where users can use both search and browsing.

5.1 Evaluating Term-based Search by Simulation

In this section, we describe our method for generating a test collection in evaluating term-based search, which is composed of documents, queries, and corresponding relevance judgments.

5.1.1 Collecting Documents

As a first step, we build a collection of documents that has the characteristics of a typical desktop. The criteria that we used for the documents in a desktop 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.

Given these conditions, our choice of a document collection method was to use the list of focus on people mentioned in the email collection from the TREC Enterprise track (crawl of the W3C website) and fetch a variety of publicly-available documents

on the web related to these people. Since the track had an expert-finding task, we had a list of people in W3C and their domains of expertise.

To describe the procedure of document collection more specifically, we 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 5.2.

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.

5.1.2 Generating Known-Item Queries

Given the collection of documents, the next step is to create queries and corresponding relevance judgments, which is usually the most time-consuming parts of building an IR test collection. However, in this case, we can generate simulated queries and relevant judgments automatically by exploiting the fact that typical requests for PIR are known-item queries [24].

5.1.2.1 Field-Based Query Generation

Although Azzopardi et al. [4] showed that generated queries can be used for retrieval experiments with web collections, a desktop collection has different characteristics, as discussed in the introduction. Among the differences, we assume that the users' querying behavior would be different for the desktop 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:
 - 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_k 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 5.2.1, 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. It would be possible to include terms outside the document in many ways, for instance by interpolating P_{term} with a collection language model, but we did not study this approach in this paper. 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.

Although there can be many variations in choosing P_{doc} and P_{field} , we use a uniform distribution that assigns equal 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. [4]

5.1.3 Evaluating Equivalence to Manual Queries

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 [4] introduced the notions of predictive and 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).

Azzopardi et al. [4] dealt with only replicative validity, but in this thesis we address 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. This means that retrieval performance comparisons will be useful only among retrieval methods whose replicative validity has been verified.

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 since the query generation procedure in general involves random selection of query terms, which in turn changes the distribution of scores. We therefore need to be interested in both measures since predictive validity is more stable but replicative validity is more strongly related to our eventual goal (retrieval results).

Lastly, we should stress that the suggested methods are not perfect measures of equivalence, since each of them tests only a particular aspect of the queries. Therefore, in the experimental section, we report the results of using hand-built queries as well as generated queries.

5.1.3.1 Verifying 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 probability $P_{term}(Q)$ of the manual query Q . This can be computed with the term distribution P_{term} from the given query generation method, as follows:

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

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.

5.1.3.2 Verifying Replicative Validity

Azzopardi et al. [4] 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 resulting p-value is greater than a certain threshold.

5.2 Pseudo-desktop Collections

Based on the method introduced above, we built pseudo-desktops collection so that it may contain typical file types in desktop like *email*, webpage (*html*) and office document (*pdf*, *doc* and *ppt*) related to specific individuals. To get the emails related to a person, we filtered the W3C mailing list collection where the name occurrence of each person was tagged [5], which enabled us to identify several individuals whose activities in W3C were prominent. For other document types, using the Yahoo! search API with the combination of name, organization and speciality of each pseudo-user as query words, we collected up to 1,000 documents for each individual and document type. In identifying the specialty of each individual, we used a list provided by TREC expert search track.

Table 5.1 lists the statistics from the resulting pseudo-desktop collections corresponding to three pseudo-users – “Jack”, “Tom” and “Kate”. Although these are prominent figures in W3C and all the collected documents are publicly available, we have anonymized their names.

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. Number and average length of documents for each pseudo-desktop collection.

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)

Table 5.2. Statistics of desktop collections from previous research.

Previous Work	#Desktops	#Files	Query Length	Document Types
Dumais et al.[20]	225	36182	1.6	e-mails: 80% / documents: 10% / others: 10%
Chernov et al.[12]	14	3433	1.7	e-mails : 82.7% / documents : 17.3% / others: 0%
Cohen et al.[14]	19	N/A	N/A	e-mails: 0% / documents: 41.2% / others: 58.8%

Table 5.2 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. These large differences further indicate the need for a reusable test collection.

Since the validation methods described in Section 5.1.3 requires hand-written queries, we collected hand-written queries for the three pseudo-desktop collections by the following procedure. 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 we described above.

5.2.1 Generated Queries

We generated queries using the set of query generation methods described in Section 5.1.2. Generated queries are verified in terms of predictive and replicative validity using some of the retrieval methods described in Section 3.1.

The result in Table 5.3 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

Table 5.3. 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

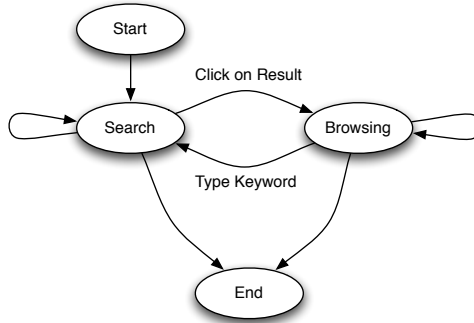


Figure 5.1. A state transition diagram of user interaction.

generation methods show replicative validity only for some of the retrieval models. Since the sample size for 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.

5.3 Summary & Proposed Work

In this chapter, we described a method for generating a reusable test collection for evaluating term-based search methods and showed the validity of queries in pseudo-desktop collections.

For the completion of this thesis, we will first improve the suggested query generation method. Currently our method takes query terms independently from a target document, and it is not realistic since actual queries would include lots of phrases, and errors, and so on. We aim to model the dependency between terms in the process of generating queries.

We will also develop a simulation-based evaluation technique for the system where term-based search and associative browsing models are combined. The basic idea is modeling user’s interaction with the system as a whole, extending the idea of modeling user’s querying behavior as presented in Section 5.1.2. The user model can be based on a state transition diagram as seen in Figure 5.1 where user start with keyword search, then use search and browsing to find the information.

This model is composed of several components: search, browsing and the transition between them. For the search component, we can use query generation method introduced in Section 5.2.1 to model which query user might use to find the target document. The rest of the model concerns the choice user can make in the form of click. At any given state, user can click on different positions of a rank list for browsing, or choose to type in another keyword for search, or choose to exit. Given this model, we can generate arbitrary sequences of user’s interaction with a system, which we can use to evaluate the effectiveness of different retrieval methods for personal information.

CHAPTER 6

GAME-BASED EVALUATION METHOD

The simulation-based evaluation method introduced in the previous chapter is 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 evaluation method has numerous benefits compared to a traditional user study. First of all, we can induce higher motivation among participants thanks to a more interesting task and the competitive nature of the game. Secondly, a good experimental control is ensured since participants are asked to complete a set of given tasks under constraints provided by the game designer. Reusability of data is another benefit considering that 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, especially when it is implemented on the server side.

One can see that game-based evaluation is not without issues, considering its artificial nature. The situation we created within a game is not the same to actual search tasks, and competitive environment may lead to unrealistic behaviors. Also, the tasks given to the users are not actual ones in the context of everyday life. Lastly, we are not using personal information that belongs to each user. However, we believe that the advantages outweigh these limitations.

In what follows, we will describe how we used game-based method for evaluating both term-based search and associative browsing model.

6.1 DocTrack Game

By adapting the PageHunt game [35] to our problem domain, we developed DocTrack game [30] for evaluating PIR methods in the context of known-item finding, as shown in Figure 6.1. In addition to using documents of many types that might be found in a desktop instead of random webpages, we made several modifications to the original PageHunt game:

First, since people generally have good knowledge of their own desktops, we collected documents that participants are familiar with and let each of them browse

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

Status

Pages Found : 2/10
Queries Issued : 0/5
Current Score : 0.0
00:01

Top Scorers

speed	100	10-01-07
speed	97	10-01-05
speed	95	10-01-08
jumpson	90	(2010-01-07)
speed	88	10-01-07
jumpson	85	(2010-01-06)
speed	84	10-01-08
jumpson	80	(2010-01-07)
sh	78	(2010-01-05)
speed	75	10-01-07
jumpson	70	(2010-01-06)

Frequent Players

speed	4 times (avg: 52)
jumpson	3 times (avg: 34)
sh	2 times (avg: 30)
jumpson	2 times (avg: 44)

Finding Self-Similarities in Opportunistic People Networks

Ling-Jyh Chen¹, Yung-Chih Chen¹, Tony Sun², Paruvelli Sreedevi¹, Kuan-Ta Chen¹, Chen-Hung Yu³, and Hao-hua Chen¹

¹Institute of Information Science, Academia Sinica
²Department of Computer Science, University of California at Los Angeles
³Department 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 network traces. 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

Figure 6.1. The screenshot of the DocTrack Search game. The user is being shown a document.

the collection for some time before starting the game; second, to simulate a typical 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.

6.1.1 CS Collection

As shown in Table 6.1, we created the CS collection by collecting emails from the Computer Science 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. The documents in the CS collection are much shorter than in the other pseudo-desktop collections.

For all document types, *title* and *content* fields were indexed. Also, there were 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, *filename* for office documents.

6.1.2 DocTrack Search Game

We first used the DocTrack game for collecting queries. Compared to the method of collecting queries described in Section 5.2, using the DocTrack game, we could gather a large quantity of realistic queries together with the whole session log data.

Table 6.1. 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 6.2. 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

This in turn allowed us to train discriminative learning models, which typically requires large amounts of data.

We had 20 participants who were students, faculty members and staff in the department, all familiar with the documents in the collection. In total, 66 DocTrack games were played and 984 queries were collected using 882 target documents, some of which are shown in Table 6.2.

The average length of queries was 3.97, which is longer than the reported length (2) in the other desktop search studies [23]. This may be due to people paying more attention to the task in the competitive game setting compared to typical desktop search. The standard deviation (1.85) of the query length was also quite high, implying that there is a considerable variation among the querying behavior of individuals.

The participants were generally in favor of the game, saying that playing the game was fun and felt reasonably similar to their search experience in the desktop. More details on retrieval experiments can be found in Section 7.1.2.2.

6.1.3 DocTrack Search & Browsing Game

For evaluating the associative browsing model introduced in Chapter 4, we designed a game in which participants can find a target document by combining keyword search and associative browsing. 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. Figure 6.2 (front) shows the interface that people used to look through the initial set of documents. The screenshot in the back of Figure 6.2 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 concept space. In the

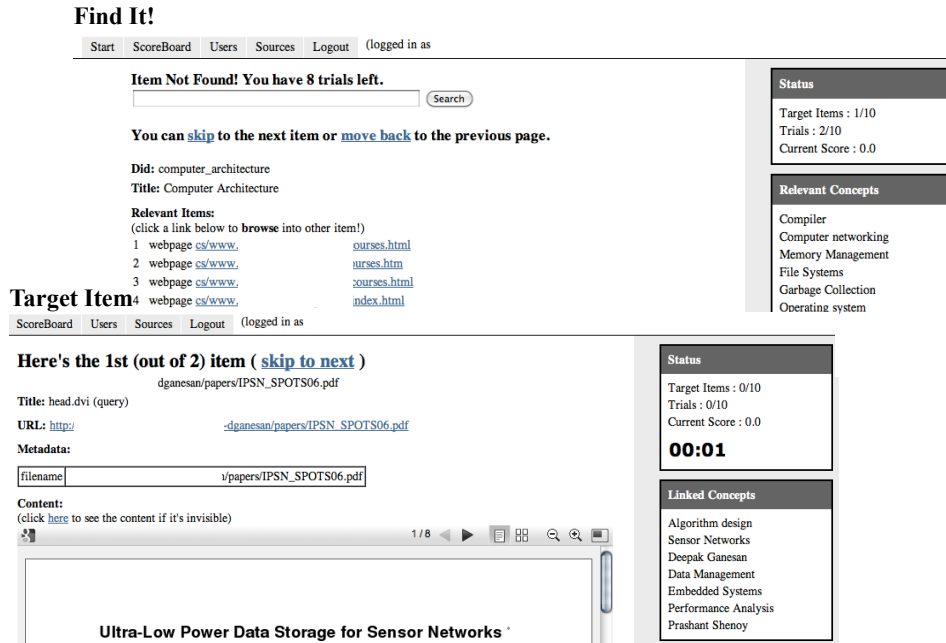


Figure 6.2. The screenshot of the DocTrack Search and Browse game. Back: the user interface for finding a target document by searching and browsing. Front: a target document is being shown along with related concepts.

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.

6.2 Summary & Proposed Work

We proposed using a game for evaluating personal information retrieval. We designed two types of games for evaluating term-based search and associative browsing, respectively. Our experiments in Chapter 7 show that one can collect a large amount of log data from many participants that can be used to train and evaluate both types of retrieval methods. Recently, we also organized a workshop ¹ based on the collection we created with this method.

As a future work, we plan to improve the game-based evaluation method. We are considering a more involved user study where we instrument a software to user's device to collect search data in a realistic setting. Although the data collected this way may suffer from privacy issues, it will reveal a valuable insight on how the data collected from our user study is different from actual search logs. And we can design a more realistic user study based on our findings.

¹Evaluating Personal Search Workshop in ECIR'11

CHAPTER 7

EVALUATION RESULTS

In this chapter we present the evaluation results of the two retrieval methods we proposed earlier – term-based search model and associative browsing. In addition to a TREC collection, test collections generated by simulation-based and game-based method were used for evaluation.

7.1 Evaluation of Term-based Search Model

In this section we describe experiments for verifying proposed term-based search model. We first describe a experiment on the type-specific retrieval performance of the PRM-S model and its variants, followed by experiments on the type prediction and final (merged) retrieval performance.

In the indexing of all collections, each word was stemmed using the Krovetz stemmer and standard stopwords were eliminated. Indri¹ was used as a retrieval engine for all the retrieval experiments. We used prediction accuracy to evaluate type prediction performance, since we have only one *correct* collection for each query. Mean Reciprocal Rank was used as the measure of retrieval performance for all experiments, since this is a known-item task where each query has only one relevant document.

7.1.1 Type-specific Retrieval Performance

Here we show the evaluation results for type-specific retrieval methods we introduced in Section 3.1 using a standard test collection. The TREC 2005 Enterprise Track known-item finding task [16] used a crawl of the W3C mailing list, containing 198,394 documents with average length of 10kb. For each document, the indexed fields were *title*, *content*, *to* (receiver), *date*, *name* (sender) 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.

Since each retrieval model required a different set of parameters to be tuned in advance, these were determined based on the effectiveness in TREC training queries. Document-level parameters such as $k1$ parameter in BM25F, interpolation ratio μ in PRM-D and Jelinek-Mercer smoothing parameter were found by parameter sweeps. For parameters that required training for each document field, such as field weights w_j and two-stage Dirichlet smoothing μ_j , we adopted a Golden Section Search algorithm.

¹<http://www.lemurproject.org>

Table 7.1. Retrieval performance for TREC known-item finding task queries.

Collection	DQL	MFLM	BM25F	PRM-S	MFLM2	PRM-D	PRM-D2	PRM-S2
TREC	0.538	0.559	0.594	0.617	<i>DM</i> 0.606	<i>D</i> 0.619	<i>DM</i> 0.624	<i>DM</i> 0.630

In Table 7.1, we compared the performance of retrieval models using the TREC test queries. Here, statistically significant (using paired t-test with p-value=0.05) improvements over baseline methods were marked using the initial character (D,M,B,P) of each baseline method.

Among the baseline methods, fixed-weight combinations of field scores such as MFLM and BM25F outperformed DQL, but not significantly. PRM-S was significantly better than DQL and PRM-D showed almost the same result as PRM-S. The application of two-stage Dirichlet smoothing resulted in significant performance gains over document-level smoothing (MFLM2) or increased the performance gap (PRM-D2 and PRM-S2).

Although none of suggested methods outperformed BM25F and PRM-S significantly, two of them (PRM-D2 and PRM-S2) were better than the best submission for the TREC 2005 Enterprise track. This result is more significant considering that all of our retrieval models are purely based on term statistics of email messages, without the aid of advanced features such as anchor-text, thread structure, and date.

7.1.2 Type Prediction and Merged Retrieval Performance

In this section we describe the experiments for verifying the type prediction and retrieval performance. 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 a computer science (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. We compared only CQL and FQL for the pseudo-desktop experiment since CQL was shown to be the most effective among collection scoring method [45] and FQL is the extension of CQL for semi-structured document collections. Section 7.1.2.2 provides the comparison with other type prediction methods using the CS collection and queries from the DocTrack game.

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).

7.1.2.1 Pseudo-desktop Collections

Three pseudo-desktop collections described in Chapter 5.1 were used for these experiments. Each collection contained typical file types such as email, webpage and

Table 7.2. Accuracy of type prediction in pseudo-desktop collections.

	Jack	Tom	Kate
CQL	0.606	0.637	0.38
FQL	0.773	0.807	0.64

Table 7.3. Retrieval performance in three pseudo-desktop collections using different type-specific retrieval methods and type prediction methods.

	Jack				Tom				Kate			
	Uniform	CQL	FQL	Oracle	Uniform	CQL	FQL	Oracle	Uniform	CQL	FQL	Oracle
DQL	0.129	0.159	0.27	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	0.15	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	0.25	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	0.32	0.212	0.212	0.349	0.477

office documents related to three individuals. More details can be found in Section 5.2.

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 some randomness.

In Table 7.2, we compare the accuracy of type prediction in pseudo-desktop collections for the CQL and FQL methods, where FQL shows a clear improvement over 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 now report the retrieval performance for the same queries in Table 7.3. 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. Especially, 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.

7.1.2.2 CS Collection

Next we report on experiments using a computer science (CS) collection, where the documents of various types are collected from many public sources in the Computer Science department the authors belong to. More details of the collection can be found in Section 6.1.1.

Since some of features required data for estimation, we used 528 queries to obtain query-log feature (QQL) values and training parameters for other features. 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,

Table 7.4. Accuracy of type prediction for best-performing single feature runs and combination methods in a CS collection.

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

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 game.

Table 7.4 summarizes the prediction accuracy result, comparing two of the best-performing single feature runs (CQL / FQL) and combination methods (Grid / RankSVM / MultiSVM). 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 combination methods. 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.

Table 7.5 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 7.5. 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	0.52
PRM-D	0.36	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	0.72	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 the big difference in experimental conditions (remember 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 is 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 aspects.

7.2 Evaluation of Associative Browsing Model

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

Table 7.6. Statistics of the sessions with search and browsing.

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

7.2.1 The Role of Associative Browsing

Here we 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.

As we can see from Table 7.6, we have 290 sessions from Round 1 and 142 sessions from Round 2. The percentage of sessions during which users chose to browse as well as search is 14% (or 42 sessions) for Round 1 and 30% (or 43 sessions) for Round 2. These significant percentages indicate that users would like to have the option of browsing in addition to search. Furthermore, the fact that we have more browsing in the second round seems to suggest that the concept space provided further motivation 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 concept as an intermediate step. Another interesting comment is that browsing was helpful for thinking of good query words.

7.2.2 Retrieval Performance

For evaluating the effectiveness of suggestions for browsing, we used three document collections to evaluate the system. 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 6.1.1. This collection is composed of 7984 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 7.7.

For learning methods, we used our own implementation of Iterative Grid Search and SVM^{rank} [25], 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 7.7. 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 7.8. 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	Uniform	Grid	SVM
Person 1	0.097	0.229	0.194	0.136	0.136	0.241	0.151	0.243	0.236	0.277
Person 2	0.037	0.350	0.403	0.221	0.310	0.516	0.234	0.506	0.581	0.509
CS/Top1	0.142	0.179	0.289	0.235	0.107	0.191	0.195	0.306	0.255	0.433
CS/Top5	0.184	0.127	0.170	0.155	0.100	0.158	0.222	0.293	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.

Here we present the evaluation results on the quality of ranking. 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, and the title and the content field was used for constructing term vectors, respectively.

Table 7.8 shows the concept ranking results for each feature and combination method. Regarding the single-feature results, different features turned out to be the most effective ones 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 exists a considerable variation 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. Another observation is that even the naive uniform combination of features produced better results than any of the ones obtained by using a feature by itself. In summary, different features perform the best for each collection, yet combination methods are consistently better than single-feature methods. All this implies that feature combination is beneficial for finding concept similarity.

As far as the document ranking task is concerned, Table 7.9 shows a slightly different trend. Term vector similarity using the **content** field is far more important

Table 7.9. 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	Uniform	Grid	SVM
Person 1	0.392	0.480	0.063	0.296	0.229	0.274	0.183	0.264	0.404	0.500	0.494
Person 2	0.334	0.564	0.268	0.372	0.184	0.137	0.092	0.187	0.512	0.592	0.478
CS/Top1	0.074	0.097	0.065	0.114	0.140	0.098	0.070	0.140	0.109	0.156	0.098
CS/Top5	0.081	0.138	0.05	0.114	0.151	0.132	0.062	0.129	0.139	0.150	0.133

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 best a priori, we can conclude that feature combination should be used here as well.

7.3 Summary & Proposed Work

In this chapter, we evaluated two retrieval methods we proposed earlier – term-based search model and associative browsing. For term-based search model, our evaluation on a TREC collection showed the performance of suggested type-specific retrieval method (PRM-S). Using the pseudo-desktop and CS collections, we demonstrated that improving the type prediction method can produce significantly better final retrieval performance. We also found that suggested type prediction method shows superior performance to competitive baseline methods in both collections we tested. Finally, Our results shows that the combination method can improve type prediction performance compared to any of existing methods.

For associative browsing, our evaluation suggests that the associative browsing model is useful for the known-item finding task, especially when concepts are used in addition to documents. We showed that the effectiveness of each link type varies according to the collection and user behavior. Furthermore, our experiments led us to the conclusion that the combination of features significantly improves the quality of suggestions.

For the completion of this thesis, we plan to verify the performance of suggested methods using other tasks and collections, since the PRM-S retrieval model and the FQL type prediction model is applicable beyond the domain of personal information as long as collection documents are structured. We hope that such investigation would reveal a valuable insight on which we can improve suggested methods further.

BIBLIOGRAPHY

- [1] Amer-Yahia, Sihem, and Lalmas, Mounia. XML search: languages, INEX and scoring. *SIGMOD Record* 35, 4 (2006), 16–23.
- [2] Arguello, Jaime, Callan, Jamie, and Diaz, Fernando. Classification-based resource selection. In *CIKM '09* (New York, NY, USA, 2009), ACM, pp. 1277–1286.
- [3] 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.
- [4] 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.
- [5] 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).
- [6] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.

- [12] 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).
- [13] Chernov, Sergey, Serdyukov, Pavel, Chirita, Paul-Alexandru, Demartini, Gianluca, and Nejdl, Wolfgang. Building a desktop search test-bed. In *ECIR' 07* (2007), pp. 686–690.
- [14] Cohen, Sara, Domshlak, Carmel, and Zwerdling, Naama. On ranking techniques for desktop search. *ACM Trans. Inf. Syst.* 26, 2 (2008), 1–24.
- [15] Craswell, Nick, and Hugo Zaragoza, Stephen Robertson. Microsoft cambridge at trec-14: Enterprise track. In *In The Fourteenth Text REtrieval Conf.* (2005).
- [16] Craswell, Nick, and Vries, Arjen P. De. Overview of the trec-2005 enterprise track. In *In The Fourteenth Text REtrieval Conf. Proc.* (2005).
- [17] Croft, W. B., and Thompson, R. H. *I³R* : A new approach to the design of document retrieval system. Tech. rep., Amherst, MA, USA, 1987.
- [18] Croft, W. Bruce, and Turtle, Howard R. Retrieval strategies for hypertext. *Information Processing and Management* 29, 3 (1993), 313–324.
- [19] Cronen-Townsend, Steve, Zhou, Yun, and Croft, W. Bruce. Predicting query performance. In *SIGIR '02* (New York, NY, USA, 2002), ACM, pp. 299–306.
- [20] Cutrell, Edward, Robbins, Daniel, Dumais, Susan, and Sarin, Raman. Fast, flexible filtering with phlat. In *CHI '06* (New York, NY, USA, 2006), ACM, pp. 261–270.
- [21] Davis, G., and Thomson, D. *Memory in context: Context in memory*. Wiley.
- [22] Dong, Xin, and Halevy, Alon Y. A platform for personal information management and integration. In *CIDR* (2005), pp. 119–130.
- [23] 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.
- [24] Elsweiler, David, and Ruthven, Ian. Towards task-based personal information management evaluations. In *SIGIR '07* (New York, NY, USA, 2007), ACM, pp. 23–30.
- [25] Joachims, Thorsten. Optimizing search engines using clickthrough data. In *KDD '02* (New York, NY, USA, 2002), ACM, pp. 133–142.
- [26] Jones, William, and Teevan, Jaime. *Personal Information Management*. University of Washington Press, 2008.

- [27] 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.
- [28] Kim, Jinyoung, Bakalov, Anton, Smith, David A., and Croft, W. Bruce. Evaluating a semantic representation for personal information. CIIR Technical Report, Univ. of Mass., Amherst.
- [29] Kim, Jinyoung, and Croft, W. Bruce. Retrieval experiments using pseudo-desktop collections. In *CIKM '09* (2009), ACM, pp. 1297–1306.
- [30] Kim, Jinyoung, 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.
- [31] Kim, Jinyoung, Xue, Xiaobing, and Croft, W. Bruce. *A Probabilistic Retrieval Model for Semi-structured Data*. In Proceedings of ECIR '09. Springer, 2009.
- [32] Lu, Chang-Tien, Shukla, Manu, Subramanya, Siri H., and Wu, Yamin. Performance evaluation of desktop search engines. In *IRI* (2007), pp. 110–115.
- [33] Lucarella, Dario. A model for hypertext-based information retrieval. 81–94.
- [34] M. Géry, C. Largeton, F. Thollard. Probabilistic document model integrating xml structure. *Proceedings in INEX* (2007), 139–149.
- [35] Ma, Hao, Chandrasekar, Raman, Quirk, Chris, and Gupta, Abhishek. Improving search engines using human computation games. In *CIKM' 09* (2009), pp. 275–284.
- [36] Metzler, Donald, and Croft, W. Bruce. Combining the language model and inference network approaches to retrieval. *IPM* 40, 5 (2003), 735–750.
- [37] Ogilvie, Paul, and Callan, Jamie. Combining document representations for known-item search. In *SIGIR '03* (New York, NY, USA, 2003), ACM, pp. 143–150.
- [38] Petkova, Desislava, Croft, W. Bruce, and Diao, Yanlei. *Refining keyword queries for XML retrieval by combining content and structure*. In Proceedings of ECIR '09. Springer, 2009.
- [39] Robertson, Stephen, Zaragoza, Hugo, and Taylor, Michael. Simple bm25 extension to multiple weighted fields. In *CIKM '04* (New York, NY, USA, 2004), ACM, pp. 42–49.
- [40] 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.

- [41] 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.
- [42] 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.
- [43] 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.
- [44] 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.
- [45] Thomas, Paul, and Hawking, David. Server selection methods in personal metasearch: a comparative empirical study. *Inf. Retr.* 12, 5 (2009), 581–604.
- [46] Tulving, E., and Thomson, D. Encoding specificity and retrieval processes in episodic memory. In *Psychological Review* (England, 1973), pp. 352–373.
- [47] von Ahn, Luis, and Dabbish, Laura. Designing games with a purpose. *Commun. ACM* 51, 8 (2008), 58–67.
- [48] Yahyaei, Sirvan, and Monz, Christof. Applying maximum entropy to known-item email retrieval. In *ECIR* (2008), pp. 406–413.
- [49] 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.