Retrieval Experiments using Pseudo-Desktop Collections

Jinyoung Kim and W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
{jykim,croft}@cs.umass.edu

ABSTRACT

Desktop search is an important part of personal information management (PIM). However, research in this area has been limited by the lack of shareable test collections, making cumulative progress difficult. In this paper, we define desktop search as a semi-structured document retrieval problem and introduce a methodology to automatically build a reusable collection (the *pseudo-desktop*) that has many of the same properties as a real desktop collection.

We then present a comprehensive evaluation of retrieval methods for semi-structured document retrieval on several pseudo-desktop collections and the TREC Enterprise collection. Our results show that a probabilistic retrieval model using the mapping relation between a query term and a document field (PRM-S) has the best performance in collections with more structure, such as email, and that the query-likelihood language model is better for other document types. We further analyze the observed differences using generated queries and suggest ways to improve PRM-S, which makes the performance gains more significant and consistent.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: [Information Search and Retrieval]

General Terms

Algorithms

Keywords

Information Retrieval, Desktop Search, Test Collection Generation, Semi-structured Document Retrieval

1. INTRODUCTION

Desktop search, which plays an important role in personal information management, has become a standard feature of most major platforms. As the amount and the complexity of information we handle every day increases, improving the effectiveness of desktop search will continue to be a significant research issue. Desktop search is different from other search applications (e.g., web search) in several key aspects: first, the collection is composed of documents of different types such as office documents, emails, presentations, and so on with different metadata associated with each type. For instance, *sender* and *receiver* will be major metadata fields for email, while *author* will be available for documents; second, queries are mostly targeted at *known items* [10]; third, major features of web collections, such as link structure and anchor text, are missing.

Past research efforts in this area have focused on the discovery of desktop-specific features [19] and desirable characteristics for user interfaces [9] [8]. This research has been limited by the lack of availability of shareable test collections. For instance, desktop search prototypes such as Stuff I've Seen [9], Phlat [8] and Connections [18] employ evaluation methods based on real users' desktop collections and queries, distributing prototype search engines to users, and evaluating performance with logged usage data. This type of evaluation is certainly valuable as it is based on actual use cases. The lack of reusability, however, makes it difficult, if not impossible, to repeat experiments and make comparisons to alternative search techniques. Another problem is that this approach requires a fully functional desktop search engine, thereby setting a high entry barrier for new researchers. In addition to the problem of test collections, there has also been a lack of clear definitions of the goals and characteristics of desktop search in comparison to other search applications.

In this paper, we define desktop search as retrieval for a semi-structured document collection with multiple schemas corresponding to each file type, reflecting the observation that desktops typically contain files with metadata and each file type has different metadata associated with it. We use the term "semi-structured" instead of "structured" because metadata in desktop environments can be loosely specified in general and is often missing, which does not fit into the traditional definition of structured data. Our definition is also relevant in the broad context of personal information management (PIM) considering that PIM increasingly encompasses more diverse items (e.g. RSS feeds, calendar events, and so on) each with a rich set of metadata.

Based on this definition, we suggest a methodology for automatically building reusable pseudo-desktop collections, consisting of document gathering and query generation. The

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'09, November 2–6, 2009, Hong Kong, China.
Copyright 2009 ACM 978-1-60558-512-3/09/11 ...$10.00.

resulting collections have many of the characteristics of typical desktop collections and, importantly, are free from the privacy concerns that are common with personal data. Our methodology includes a novel query generation method and a technique for comparing the generated queries with real queries.

Using our approach, we present retrieval experiments for a model that extends past work on structured document retrieval and compare it against state-of-the-art retrieval models. Based on three pseudo-desktop collections as well as the TREC Enterprise collection, our evaluation shows that a retrieval model that maps query terms to document fields (PRM-S) has the best performance for email collections, whereas the query-likelihood language model is better for other document types.

Since the performance differences were not consistent, however, we further analyse the performance characteristics of PRM-S and suggest how it can be improved by better estimation of field-level scores. Again, experimental results show that this modification makes the performance gains more significant and consistent in many cases.

Our approach to desktop search has its limits, in that some elements of the desktop are missing in the pseudo-desktop. For instance, metadata such as the folder hierarchy, file creation date and usage log are not available for a generated desktop collection. However, we found equivalent features for some metadata fields, and past work in related fields [5] [21] showed that missing features can be independently developed and incorporated into the retrieval model subsequently.

Also, we cannot claim that a generated test collection is an ideal substitute for a real desktop environment with actual user queries, considering that it is impossible to simulate all aspects of document gathering and query formulation by users. Instead, we tried to make the collection generation procedure as realistic as possible, and verify the validity of the resulting test collection for retrieval experiments by comparison to actual instances of desktops and user queries.

The rest of this paper is organized as follows. In the next section, we provide an overview of related work. Then we introduce our test collection generation method and describe how we can demonstrate equivalence to real collections. We then describe existing retrieval models for desktop search and possible improvements to the PRM-S model. In the experiments, we report retrieval results using a well-studied TREC collection (from the Enterprise track) and several pseudo-desktop collections generated using the suggested method.

2. RELATED WORK

Related work can be found in three interconnected but different areas: semi-structured document retrieval, known-item search, and desktop search. They are connected in a sense that each of them characterizes the problem in terms of the document, the query and the environment, but they differ in the emphasis of the research. The first area focuses mainly on the retrieval model, whereas research efforts in the other two areas have concentrated more on task-specific features and evaluation methods.

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 [16]. This straightforward combination of field-level scores was found to have limitations, resulting in efforts such as BM25F [17]. Recently, an adaptation of score combination and smoothing method was suggested [23] for the language modeling approach to IR, based on the search engine Indri [15] which supports combining evidence from multiple fields.

The TREC 2005 Enterprise Track [7] provided a known-item email retrieval task, where a set of emails and corresponding queries were given. Among the participants, the BM25F model [6] combined a variety of document fields and other features such as the year and the thread structure to get good effectiveness. Another approach [21] combined different independent sources to improve the performance of known-item search.

Desktop search systems such as Stuff I've Seen [9] and Phlat [8] showed that user interaction is a significant issue in the desktop environment, and that *date* is most important feature since most users sorted the results by date. Other researchers focused more on improving the quality of ranking and proved that temporal locality and causality [18] are useful features. Learning feature weights with training data [5] has also been found to be effective in the desktop environment. The approach of treating desktop search as meta-search problem has been suggested [20], although the focus of this research was the characterization and selection of servers, while we focus here on the effective retrieval of individual collections.

Evaluation of desktop search or, in general, personal information management (PIM), has been considered a tricky problem [11] because real desktop collections are not available due to privacy concerns. Performance evaluation of major commercial desktop search engines was tried [14] in standard IR evaluation settings, using TREC Robust track data. Chernov et al [4] [3] proposed a method for creating a testbed for desktop search by collecting documents and queries collaboratively, yet no experimental validation was done. Elsweiler [10] suggested an evaluation method for PIM based on user studies. The approach described in this paper is different in that it does not require any direct user involvement.

3. GENERATING A PSEUDO-DESKTOP

In this section, we describe our method for generating a pseudo-desktop test collection, which is composed of documents, queries, and corresponding relevance judgments.

3.1 Collecting Documents

As a first step, we need a collection of documents that has the characteristics of a typical desktop. The criteria that we used for the documents in a desktop were that the documents should be related to a particular person, there should be of a variety of document types, the different document types should have metadata or fields, and that the collection should be of reasonable size, although there is no hard limit on size since real-world desktops vary considerably. [8] The privacy of the target individual was another concern.

Given these conditions, our choice of a document collection method was to 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 those people. More specifically, we filtered the existing mailing list and webpage from the collection to get emails which refer to a set of 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 5.2.1.

In addition to satisfying the conditions above, this method provides 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 the rich set of metadata provided by a web search engine together with the documents. For instance, for the web search engine we used, the document title, URL, and the abstract were available.

3.2 Generating Known-Item Queries

Once we have the collection of documents, the next steps are creating queries and corresponding relevance judgments, which are 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 desktop search are known-item queries [10].

3.2.1 Document-Based Query Generation

Given a situation where a user is trying to find a document that she has seen (or created) previously, she may try to come up with whatever terms she can remember from the document. Based on this observation, Azzopardi et al. [1] suggested a set of methods for generating a known-item query by algorithmically selecting a set of terms from the target document, as illustrated below.

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

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

3.2.2 Field-Based Query Generation

Although Azzopardi et al. [1] 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 desktop search 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 the experimental section, 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. [1]

3.3 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 [1] 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. [1] dealt with only replicative validity, but in this paper 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.

3.3.1 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:

$$P_{term}(Q) = \prod_{q_i \in Q} P_{term}(q_i) \tag{1}$$

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.

3.3.2 Verifying Replicative Validity

Azzopardi et al. [1] 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.

4. RETRIEVAL MODELS FOR DESKTOP SEARCH

In this section we introduce retrieval models for desktop search. We explain existing retrieval models first. Then we suggest improvements over existing methods. The following notation will be used throughout this paper. We assume that a query $Q = (q_1, ..., q_m)$ is composed of m words, and the collection C contains n field types $(F_1, ..., F_n)$. Each document d in the collection may include fields $(f_1, ..., f_n)$, where each field is marked using lowercase letters to distinguish it from the corresponding field type in the schema. We also denote field-level weights $(w_1, ..., w_n)$. For retrieval models that require field-level smoothing parameters, $(\mu_1, ..., \mu_n)$ was used. Lastly, $|f_k|$ means the length of field f_k in words, $tf(t, f_k)$ denotes the term frequency of t in field f_k and $df(t)$ denotes the document frequency of t. Model-specific parameters will be explained as they appear.

4.1 Existing Retrieval Models

4.1.1 BM25F

BM25F [17] is the modification of the BM25 model where field-level evidence is combined at the raw frequency level rather than score level. This maintains non-linear saturation of term frequencies. The BM25F score S is calculated as:

$$S(q, d) = \sum_{q_i \in Q} idf(q_i) \frac{weight(q_i, d)}{k_1 + weight(q_i, d)} \tag{2}$$

where term weight $weight(q_i, d)$ is calculated as:

$$weight(q_i, d) = \sum_{f_j \in d} \frac{w_j tf(q_i, f_j)}{(1 - b_j) + b_j \frac{|f_j|}{|F_j|}} \tag{3}$$

Here, $|F_j|$ denotes the average length of field f_j across the whole collection, and field-level parameter b_j controls the degree of length normalization and is tuned using training queries.

4.1.2 Mixture of Field Language Models

Ogilvie et al. [16] suggested a mixture of field language models by linear interpolation (MFLM) for known-item search. A document score in the MFLM is calculated by taking a weighted average of field-level scores as follows:

$$P(Q|d) = \prod_{i=1}^{m} \sum_{j=1}^{n} w_j P_{QL}(q_i|f_j) \tag{4}$$

$P_{QL}(q_i|f_j)$ is the query-likelihood score of field f_j after appropriate smoothing with the background field language model F_j using Jelinek-Mercer smoothing [22] with parameter λ_{JM}. The weight parameters w_j are tuned with training queries, but do not vary for different queries. This can be too restrictive if each field can provide a different strength of evidence for different query terms. The next model relaxes this restriction by inferring the relationship between query terms and document fields.

4.1.3 Probabilistic Retrieval Model for Semi-structured Data

The probabilistic retrieval model for semistructured data (PRM-S) [12] is the extension of MFLM in a sense that it also combines field-level scores into a document-level score. The difference compared to MFLM is that the mapping between query terms and document fields are used as weights, which vary for each query term. We can infer this mapping between each query term and document field based on collection 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 $P_M(w|F_j)$.

$$P_M(F_j|w) = \frac{P_M(w|F_j) P_M(F_j)}{\sum_{F_k \in F} P_M(w|F_k) P_M(F_k)} \tag{5}$$

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:

$$P(Q|d) = \prod_{i=1}^{m} \sum_{j=1}^{n} P_M(F_j|q_i) P_{QL}(q_i|f_j) \tag{6}$$

4.2 Improving PRM-S

Past work [12] showed that PRM-S has significant performance advantages for retrieval in some semi-structured data collections. Here we investigate some potential improvements. As PRM-S is the combination of two elements – the mapping probability $P_M(F_j|q_i)$ and field-level scores $P_{QL}(w|f_j)$ – we can divide the task into improving the estimation of either element. Mapping probability estimation is not addressed in this paper. For the task of field-level score estimation, the first issue is whether the field-level query likelihood score is an appropriate term-weighting function. Another issue is whether the core assumption of PRM-S that each query term is chosen from a specific document field is correct. Some users may choose query terms without considering the fields, thereby making it inappropriate to use the mapping probability for weighting.

In our current work, we tried to address both issues. Specifically, we attempted to improve the current estimation by the following methods.

4.2.1 Two-stage Dirichlet Smoothing

Zhao et al. [23] suggested two-stage Dirichlet smoothing. Here, each document is smoothed by the collection language model C first, then this smoothed document language model is used to smooth the field language model.

$$P_{QL}(q_i|f_j) = \frac{|f_j|}{|f_j| + \mu_j} P(q_i|f_j) + \frac{\mu_j}{|f_j| + \mu_j} P(q_i|d)' \tag{7}$$

$$P(q_i|d)' = \frac{|d|}{|d| + \mu} P(q_i|d) + \frac{\mu}{|d| + \mu} P(q_i|C) \tag{8}$$

They argued that this modification would address the problem of unbalanced field-level smoothing where the term occurrences in a shorter field have more impact than those in a longer field. However, this method requires field-level smoothing parameters μ_j to be tuned as well as document smoothing parameter μ, which adds complexity to the previous model [22].

4.2.2 Mixture of PRM-S and Document Query Likelihood

As we pointed out above, it may not make sense to assume that every query term is generated with a particular field in mind. In such cases, 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 9, thereby striking a balance between these two.

$$P(Q|d) = \prod_{i=1}^{m} ((1-\lambda) \sum_{j=1}^{n} P_M(F_j|q_i) P_{QL}(q_i|f_j) + \lambda P_{QL}(q_i|d)) \tag{9}$$

where λ is the parameter that controls the interpolation ratio. This model bears similarity to two-stage Dirichlet smoothing in that it combines document language model with field-language model.

5. EXPERIMENTS

In this section we describe the experiments comparing the retrieval effectiveness of the retrieval models and validating the test collection generation method. We used the TREC 2005 Enterprise track collection for the initial experiments. We also report on experiments using three pseudo-desktop collections. We describe the collection generation procedure and then the retrieval experiments using both hand-written queries and generated queries.

For indexing both collections, each word was stemmed using the Krovetz stemmer and standard stopwords were eliminated. 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. Indri[1] was used as a retrieval engine, which required some modification to accommodate BM25F scoring.

The retrieval models compared were the document language model (DLM), BM25F, the mixture of field language models (MFLM), the probabilistic retrieval model for semistructured data (PRM-S) and the linear interpolation of PRM-S and DLM (PRM-D). We also applied two-stage smoothing for retrieval models based on the field language model – MFLM (MFLM2), PRM-S (PRM-S2) and PRM-D (PRM-D2).

5.1 TREC Enterprise Collection

The TREC 2005 Enterprise Track known-item task [7] 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.

5.1.1 TREC Queries

In Table 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 DLM, but not significantly. PRM-S was significantly better than DLM 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-

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

Table 1: Retrieval performance for TREC test queries.

Collection	DLM	MFLM	BM25F	PRM-S	MFLM2	PRM-D	PRM-D2	PRM-S2
TREC	0.538	0.559	0.594	0.617	$_{DM}$0.606	$_{D}$0.619	$_{DM}$0.624	$_{DM}$0.630

Table 2: Sum of generation probabilities (in log) for different generation methods on TREC queries. Field-based generation method with TF*IDF-based term selection show highest probability.

Extent : Document		Extent : Field	
P_{term}	$P_{term}(Q)$	P_{term}	$P_{term}(Q)$
Uniform	-26.457	Uniform	-23.460
TF	-22.782	TF	-22.394
IDF	-21.876	IDF	-17.990
TF*IDF	-18.269	TF*IDF	-17.180

Table 3: P-values of Kolmogorov-Smirnov test for different query generation methods on TREC queries. Queries generated with field-based method have higher p-values in overall.

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

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.

5.1.2 Generated Queries

Next we report experiments using query generation methods with the TREC data. The query generation methods we tested varied by the structure from which query terms were selected (document vs. field), and the type of probability distribution P_{term} from which query-terms were selected (uniform / TF / IDF / TF*IDF). For P_{length}, we used the length distribution of actual TREC queries. Since all query generation methods we used included some randomness in the process, we repeated all experiments three times and report the averaged result.

We first report the verification result of predictive validity using the generation probabilities of 150 TREC queries. Table 2 shows that field-based query generation methods have higher probabilities of generating manual queries than document-based generation methods, although it does not provide an absolute criteria for judging the equivalence to TREC queries. Among term selection methods, the term selection by TF*IDF was shown to have higher probability for generating TREC queries, reflecting users' behavior of choosing popular and discriminative terms.

We then evaluated each query generation method in terms of replicative validity using the Kolmogorov-Smirnov (KS) test, as seen in Table 3. Each cell contains the average p-value of the KS-test for the score distributions of TREC queries and generated queries using corresponding method.

We report only the results of DLM, PRM-S and PRM-D because they were best-performing models for which replicative validity was established for any of query-generation methods. For BM25F and MFLM, none of suggested methods succeeded and two-stage Dirichlet smoothing runs were omitted since they were slight variants of other retrieval models. The result shows that field-based query generation methods have higher p-values in general, and uniform or TF-based term selection methods have p-values greater than the threshold (0.05) for all retrieval models tested.

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

The results in Table 3 support this explanation since the field-based generation method with the TF*IDF term selection has the highest replicative validity for DLM yet fails on PRM-S and PRM-D, which incorporate the mapping probability into the field-level query likelihood score. This is also consistent with the results of Azzopardi et al. [1] where TF*IDF term selection method was shown to have highest replicative validity for document-based retrieval models. Despite this, we can conclude that field-based generation is more valid for use in semi-structured document retrieval experiments, since it was shown to have both higher predictive and replicative validity.

Next, we report the retrieval performance based on generated queries in Table 4, where the relative performance of each retrieval model for different query generation methods reveal interesting trends. Note that this result is equivalent to hand-built queries only for retrieval models (DLM, PRM-S, PRM-D) for which replicative validity was shown above.

The most conspicuous trend here is that the relative performance of DLM and PRM-S are different for document-based vs. field-based query generation methods. While DLM is better than any other retrieval model for document-based query generation methods, it is worse than PRM-S and its variants when field-based query generation was used. Note also that PRM-D shows the best performance regardless of the query generation method.

Another interesting observation is that the impact of two-stage Dirichlet smoothing varies depending on which retrieval model it is used with. When applied to MFLM (MFLM2), two-stage Dirichlet smoothing improved the performance considerably, resulting in better performance than BM25F. For PRM-S, it produced performance gains only for document-based query generation methods. On the other hand, it hurt the performance of PRM-D, which makes sense

Table 4: Retrieval performance for generated queries in TREC collection.

Extent	P_{term}	DLM	MFLM	BM25F	PRM-S	MFLM2	PRM-D	PRM-D2	PRM-S2	Average
Document	Uniform	0.646	0.137	0.219	0.346	0.356	0.578	0.476	0.444	0.376
	TF	0.588	0.168	0.292	0.429	0.390	0.558	0.515	0.491	0.405
	IDF	0.875	0.288	0.431	0.65	0.551	0.812	0.761	0.713	0.615
	TF*IDF	0.887	0.317	0.537	0.66	0.566	0.836	0.770	0.745	0.652
Average		0.749	0.228	0.369	0.521	0.466	0.696	0.630	0.598	0.512
Field	Uniform	0.461	0.134	0.299	0.629	0.338	0.631	0.631	0.633	0.444
	TF	0.452	0.124	0.256	0.584	0.292	0.596	0.578	0.577	0.410
	IDF	0.619	0.180	0.424	0.730	0.454	0.728	0.711	0.721	0.561
	TF*IDF	0.647	0.222	0.462	0.75	0.429	0.739	0.743	0.738	0.581
Average		0.545	0.165	0.360	0.673	0.378	0.674	0.666	0.667	0.499

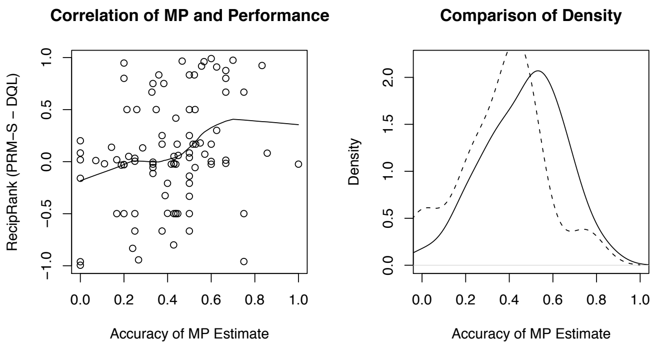

Figure 1: Left : Correlation of retrieval performance (Reciprocal Rank) and the accuracy of mapping probability (MP) estimation. Right : Comparison of MP distribution between queries where PRM-S is better (solid line) vs. DLM is better.

considering that this model already incorporates the document language model.

5.1.3 Analysis of the Performance

Experiments on the TREC collection confirmed that PRM-S is the best among the baseline methods and the combination with DLM improves the performance further, which seems to suggest that PRM-S uses a different notion of relevance which complements the document language model score. An example TREC query where PRM-S was better than DLM gives some insight on the performance characteristics of PRM-S. For the query 'SWSL telecon feb 2004', PRM-S and its variants were better than any other baseline methods. Here, all query terms were correctly mapped into the right elements (*title*, *title*, *date*, *date*) with very high probabilities (>0.9). However, the use of the query-field mapping can be a problem since PRM-S was worse than other baseline methods for queries like 'Amazon Access How Accessible', where the query terms were predicted to be in the *title* field, yet were found in *content*.

To generalize the findings from these examples, we did an analysis of what affects the relative performance gain of PRM-S compared to DLM, which represent the best-performing field-based and document-based retrieval models, respectively.

Since we first hypothesized that what makes PRM-S perform better than DLM would be the accuracy of mapping probability (MP), we plotted the difference in retrieval scores of two models against the accuracy of MP estimation – in what portion of query-words MP predicted the field correctly. In left side of Figure 1, there is a weak correlation (Pearson correlation coefficient = 0.27) between the accuracy of MP estimation and the difference in Reciprocal Rank of PRM-S and DLM. In the right side of the same figure, we compared the kernel density estimation of MP accuracy for two query sets for which either PRM-S or DLM performs better, where we find that queries for which PRM-S outperforms have higher mapping probabilities in general. The means of these two densities are also significantly different, confirmed by two-sided t-test (p-value < 0.05).

To verify whether the choice of field during query formulation has an impact on the performance advantage of PRM-S, we examined the location in the relevant documents the most query terms were found. We found that 50% of the queries where PRM-S outperformed DLM had *title* as the field containing the most query terms, while the most frequent field was *content* for 67% of queries for which DLM outperformed PRM-S. In Section 5.1.3, we further verify this observation using generated queries.

5.2 Pseudo-Desktop Collections

Next we report the experiments using three pseudo-desktop collections we generated. We first describe how pseudo-desktops were built based on W3C collection and web queries. Then we show the retrieval performance using hand-written queries, followed by the results with generated queries. For retrieval experiment in generated queries we used only DLM, PRM-S and PRM-D, for which we have shown that replicative validity was verified.

5.2.1 Building a Pseudo-desktop Collection

As described in Section 3, we built each pseudo-desktop 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 [2], 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. Indexed fields were the same as the TREC collection for *email*. For other document types, *title*, *URL*, *abstract*, *date*, *text* were indexed.

Table 5 lists the statistics from the resulting pseudo-desktop

Table 5: 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)

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 6 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 more reusable test collection.

5.2.2 *Hand-Written Queries*

We have previously shown that field-based query generation methods have replicative validity to hand-written TREC queries. To verify this again on the pseudo-desktop collections, 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.

Table 7 shows the retrieval results for the hand-written queries, where we also used the same set of trained parameters for this experiment since they were subsets of TREC collection. Here, statistically significant improvements over baseline methods were marked using the initial character (D,M,B,P) of each baseline method.

Similarly to the experiments with generated queries and the TREC collection, there exists considerable variations in absolute and relative performance between DLM and PRM-S, with DLM performing better for Jack and Kate, while PRM-S is better for Tom. Still, PRM-D outperforms both methods for Jack's and Tom's and performs almost as well as DLM for Kate's, showing the same performance characteristics found as in the TREC collection.

The impact of two-stage smoothing is re-confirmed here, where it helped MFLM and PRM-S yet hurt the performance of PRM-D. Lastly, BM25F and MFLM are not as good as DLM for this collection. Given that these methods have two sets of field-level parameters and all parameters were trained using TREC queries with very similar characteristics to this one, this seems to suggest high sensitivity of these models to parameter tuning.

5.2.3 *Generated Queries*

We generated queries using the same set of query generation methods to the TREC collection. Generated queries are verified in terms of predictive and replicative validity

Table 8: P-values of Kolmogorov-Smirnov test for different query generation methods in pseudo-desktop collections.

Extent	P_{term}	DLM	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

using the same method to the TREC collection. The result in Table 8 shows the same trends as the TREC collection, reconfirming the replicative validity of field-based generation methods, especially when query-terms were selected randomly or based on term frequency. Document-based generation methods show replicative validity only for some of the retrieval models. Since the sample size for 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.

We next show the retrieval results for pseudo-desktop collections, where we generated 100 queries of length 2, following the average query length of previous works. The field-based method was used for query generation since it achieved replicative validity in experiments using this collection as well as the TREC collection. For term selection probability P_{term}, we used both uniform and TF-based selection since they showed replicative validity in both TREC and pseudo-desktop collections. Here, we report the result only on TF-based term selection as uniform selection showed the same trend.

Table 9 shows the retrieval results for each collection type and user. Different retrieval model performed best in different sub-collections: PRM-S in the *email* collection, DLM in *doc* and *ppt*. PRM-D performed the best only in the *html* collection, yet its performance was close to the best in most cases.

The result shows that the relative performance of each retrieval method depends on the type of sub-collection, although there are considerable variations within the same type. PRM-S has the advantage in the *email* collection where each document is composed of many fields with different term distributions, while DLM seem to have advantages in more traditional document collections such as webpages and office documents where each field may contain similar words. This result is consistent with previous work [12] where the performance advantage of PRM-S was greater in a collection with a clear field semantics and therefore the estimation of query-field mapping is easier.

5.2.4 *Analysis of the Performance*

To support the claim above, we performed a similar analysis as in Section 5.1.3 to find out the correlation between the accuracy of mapping probability (MP) estimate and the relative performance of DLM, PRM-S and PRM-D. Here, each of 45 data points (3 users × 5 sub-collections × 3 repetitions) represents an unit experiment.

In both sides of Figure 2, we can find rather strong correlation between the accuracy of MP estimate and the difference

Table 6: Statistics of desktop collections from previous research.

Previous Work	#Desktops	#Files	Query Length	Document Types
Dumais et al.[8]	225	36182	1.6	e-mails: 80% / documents: 10% / others: 10%
Chernov et al.[3]	14	3433	1.7	e-mails : 82.7% / documents : 17.3% / others: 0%
Cohen et al.[5]	19	N/A	N/A	e-mails: 0% / documents: 41.2% / others: 58.8%

Table 7: Retrieval performance for hand-built queries in pseudo-desktop collections.

Collection	DLM	MFLM	BM25F	PRM-S	MFLM2	PRM-D	PRM-D2	PRM-S2
PD-Jack	0.378	0.235	0.229	0.334	$_{M}$0.301	$_{MBP}$0.389	$_{MBP}$0.341	$_{MB}$0.356
PD-Tom	0.403	0.312	0.311	0.422	0.362	$_{MB}$0.457	$_{MB}$0.435	$_{MB}$0.438
PD-Kate	0.482	0.307	0.401	0.413	$_{M}$0.361	$_{MP}$0.463	$_{M}$0.453	$_{MP}$0.455

Table 9: Retrieval performance for generated queries in pseudo-desktop collections. Queries are generated using field-based method with random term selection.

Type	User	DLM	PRM-S	PRM-D
email	Jack	0.213	0.285	0.275
	Tom	0.197	0.244	0.243
	Kate	0.329	0.438	0.413
Average		0.246	0.323	0.310
html	Jack	0.418	0.428	0.440
	Tom	0.428	0.381	0.409
	Kate	0.422	0.377	0.410
Average		0.423	0.395	0.419
pdf	Jack	0.417	0.378	0.410
	Tom	0.411	0.359	0.392
	Kate	0.408	0.420	0.453
Average		0.412	0.385	0.419
doc	Jack	0.407	0.222	0.284
	Tom	0.359	0.502	0.511
	Kate	0.538	0.341	0.398
Average		0.480	0.350	0.398
ppt	Jack	0.412	0.221	0.286
	Tom	0.451	0.555	0.569
	Kate	0.468	0.397	0.440
Average		0.444	0.391	0.432

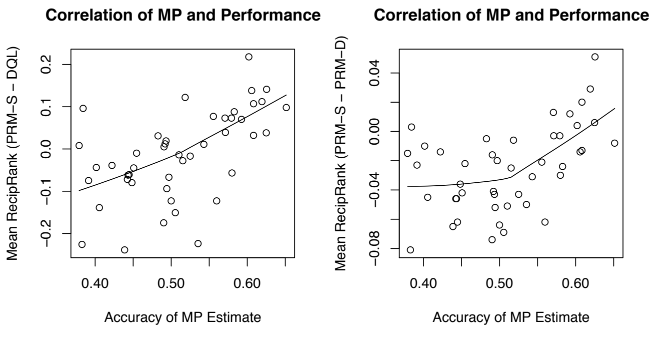

Figure 2: Correlation of the aggregate retrieval performance (mean Reciprocal Rank) and the average accuracy of mapping probability (MP) estimation. Left : difference between PRM-S and DLM. Right : difference between PRM-S and PRM-D.

Table 10: Retrieval performance for generated queries with different field selection. For instance, *title* means that query-terms were chosen only from the title field of document.

Type	Field	DLM	PRM-S	PRM-D
email	title	0.251	0.389	0.342
	content	0.339	0.267	0.327
Average		0.295	0.328	0.335
html	title	0.461	0.533	0.547
	content	0.514	0.278	0.339
Average		0.488	0.406	0.443

in the MAP of PRM-S and DLM (left), PRM-S and PRM-D (right). The Pearson correlation coefficient is 0.57 on the left side and 0.50 on the right side, which means that the performance advantage of PRM-S over DLM and PRM-D is correlated with the accuracy of MP estimate. While this is consistent with the observation in Figure 1 except that the correlation is stronger here, we can infer that the use of aggregate statistics would have reduced randomness and increased the correlation.

Lastly, to confirm the claim in Section 5.1.3 that the performance advantage of PRM-S is correlated with the choice of field from which query terms are taken, we did another retrieval experiment with generated queries where query terms are taken from only one field (*title* or *content*). The result matched our expectation that PRM-S and PRM-D will have advantages when query terms are chosen from the *title* field as opposed to the *content* field, with a larger performance gap for the *email* collection as seen in Table 10.

6. CONCLUSION

In this paper, we described a method for generating a reusable test collection for desktop search experiments, showed that the pseudo-desktop collections are valid based on various criteria, and reported the results of retrieval experiments using the pseudo-desktop collections and the TREC Enterprise collection. In addition to the verification of replicative validity, we suggested how we can verify predictive validity using the generation probabilities of manual queries. The retrieval experiments compared a number of retrieval models designed for search with semi-structured data that is typical of a desktop collection. We suggested several modifications for better estimation of field-level scores of the PRM-S model and analyzed the performance of the different retrieval methods using generated queries of varying characteristics.

Our experimental results with the TREC and pseudo-desktop collections suggest that the performance advantage of PRM-S model is dependent on the accuracy of the mapping probability estimation and the characteristics of document fields that the query terms are taken from. Also, in all experiments with hand-built queries, the interpolation of PRM-S and the document language model gives more consistent and significant performance gains regardless of the relative performance of these baseline methods. Two-stage Dirichlet smoothing addresses the same point yet is not as robust as this simple interpolation model, showing high-sensitivity on field-level smoothing parameters. This was also observed for field-based retrieval models such as BM25 and MFLM, which require the training of field-level parameters.

Our work leaves many interesting challenges. Since this is the first attempt in building reusable desktop collections, we can refine the generation procedures using more sophisticated query generation models or scale the collection by adding more file types and metadata fields. For instance, the known-item query generation methods suggested in this paper can be made more realistic by including phrases as well as words.

Concerning the retrieval models, it was found that the accuracy of mapping probability (MP) estimation has a substantial impact on the performance advantage of PRM-S. Since PRM-S only considers field-level collection statistics to estimate mapping probability, it can be improved by using more elaborate estimation techniques that takes more features into account, as recently demonstrated by Li et al.[13]

Also, as we dealt with only sub-collection level retrieval in this paper, we need to find how these results can be incorporated into a single rank list effectively. Since our experiments suggest that each retrieval model has advantages in different sub-collections, we can approach the problem by retrieving each sub-collection with the most suitable method and then merging the results appropriately.

7. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0707801, and NSF grant #IIS-0711348. Any opinions, findings and conclusions or recommendations expressed are the authors' and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR '07*, pages 455–462, New York, NY, USA, 2007. ACM.

[2] S. Bao, H. Duan, Q. Zhou, M. Xiong, Y. Cao, and Y. Yu. Research on expert search at enterprise track of trec 2006. In *15th Text REtrieval Conf.*, 2006.

[3] S. Chernov, G. Demartini, E. Herder, M. Kopycki, and W. Nejdl. Evaluating personal information management using an activity logs enriched desktop dataset. In *Personal Information Management Workshop at CHI 2008*, April 2008.

[4] S. Chernov, P. Serdyukov, P.-A. Chirita, G. Demartini, and W. Nejdl. Building a desktop search test-bed. In *ECIR*, pages 686–690, 2007.

[5] S. Cohen, C. Domshlak, and N. Zwerdling. On ranking techniques for desktop search. *ACM Trans. Inf. Syst.*, 26(2):1–24, 2008.

[6] N. Craswell and S. R. Hugo Zaragoza. Microsoft cambridge at trec-14: Enterprise track. In *In The Fourteenth Text REtrieval Conf.*, 2005.

[7] N. Craswell and A. P. D. Vries. Overview of the trec-2005 enterprise track. In *In The Fourteenth Text REtrieval Conf. Proc.*, 2005.

[8] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin. Fast, flexible filtering with phlat. In *CHI '06: Proceedings of the SIGCHI conference*, pages 261–270, New York, NY, USA, 2006. ACM.

[9] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: a system for personal information retrieval and re-use. In *SIGIR '03*, pages 72–79, New York, NY, USA, 2003. ACM.

[10] D. Elsweiler and I. Ruthven. Towards task-based personal information management evaluations. In *SIGIR '07*, pages 23–30, New York, NY, USA, 2007. ACM.

[11] W. Jones and J. Teevan. *Personal Information Management*. University of Washington Press, 2008.

[12] J. Kim, X. Xue, and W. B. Croft. *A Probabilistic Retrieval Model for Semi-structured Data*. In Proceedings of ECIR '09. Springer, 2009.

[13] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR '09*, New York, NY, USA, 2009. ACM.

[14] C.-T. Lu, M. Shukla, S. H. Subramanya, and Y. Wu. Performance evaluation of desktop search engines. In *IRI*, pages 110–115, 2007.

[15] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *IPM*, 40(5):735–750, 2003.

[16] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference*, pages 143–150, New York, NY, USA, 2003. ACM.

[17] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *In Proceedings of CIKM '04*, pages 42–49, New York, NY, USA, 2004. ACM.

[18] S. Shah, C. A. N. Soules, G. R. Ganger, and B. D. Noble. Using provenance to aid in personal file search. In *ATC'07: 2007 USENIX*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.

[19] C. A. N. Soules and G. R. Ganger. Connections: using context to enhance file search. In *SOSP*, pages 119–132, 2005.

[20] P. Thomas. Server characterisation and selection for personal metasearch. 2008.

[21] S. Yahyaei and C. Monz. Applying maximum entropy to known-item email retrieval. In *ECIR*, pages 406–413, 2008.

[22] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[23] L. Zhao and J. Callan. A generative retrieval model for structured documents. In *In Proceedings of CIKM '08*, pages 1163–1172, New York, NY, USA, 2008. ACM.