Evaluating Search in Personal Social Media Collections

Chia-Jung Lee, W. Bruce Croft and Jin Young Kim
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts, Amherst
{cjlee, croft, jykim}@cs.umass.edu

ABSTRACT

The prevalence of social media applications is generating potentially large personal archives of posts, tweets, and other communications. The existence of these archives creates a need for search tools, which can be seen as an extension of current desktop search services. Little is currently known about the best search techniques for personal archives of social data, because of the difficulty of creating test collections. In this paper, we describe how test collections for personal social data can be created by using games to collect queries. We then compare a range of retrieval models that exploit the semi-structured nature of social data. Our results show that a mixture of language models with field distribution estimation can be effective for this type of data, with certain fields, such as the name of the poster, being particularly important. We also analyze the properties of the queries that were generated by users with two versions of the games.

Categories and Subject Descriptors

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

General Terms

Design, Performance, Experimentation

Keywords

Personal social media collections, Facebook, Twitter, search evaluation, semi-structured document, desktop search

1. INTRODUCTION

The growing popularity of social media applications such as Facebook, Twitter and Google+ has generated huge archives of potentially useful information. From an individual's point of view, these social media services provide frequent updates, and considerable personal archives of posts, tweets, and other communications can accumulate. The existence of these personal archives creates a potential search problem that is not addressed by global search mechanisms such as Twitter search[1] and Facebook search[2]. In this paper, we focus on studying the personal search tools that can be viewed as an extension of current desktop search services.

Research on desktop search has focused on the problems of developing test collections[6], retrieval models, and evaluation methods. Despite this research, results are currently only known for more "conventional" document types such as email or office documents; information about search techniques for personal social archives is quite limited. To understand this new type of personal search, we explore and evaluate a range of search techniques on test collections created using personal archives from social applications. In particular, we focus on the task of known-item search on personal social archives, as known-item queries are the most frequent type of request in a desktop environment[5]. Moreover, according to Teevan et al [15], people have reported that attempting to re-find previously encountered information with global Twitter archives usually ended in failure, showing that this is both a real and difficult problem.

Another property of social media data is the rich set of fields associated with the content, which is usually presented in the form of semi-structured documents. Exploiting this structure can be helpful for improving retrieval performance as well as understanding field characteristics. Several field-based retrieval techniques [12][13][6] have been proposed to deal with semi-structured text for typical desktop data. The common principle among these approaches is to estimate the weights representing the query intent over structural components for query terms. Specifically, the probabilistic retrieval model for semi-structured data (PRMS) [6] has reported effectiveness improvements based on per-term weight estimation. However, little is known about suitable retrieval models for social media data, which has some significant differences such as document length compared to typical desktop data.

In this paper, we propose a method for building test collections from personal social media data. We collect personal archives from Facebook or Twitter as the document collections. Two types of human computation games, DocTrack[7] and MemRecap, are proposed for collecting known-item queries from real users. Based on the test collections, we are able to compare a range of retrieval models that exploit the semi-structured nature of social data. Our results show that a mixture of language models with field distribu-

WSDM'12, February 8–12, 2012, Seattle, Washington, USA.

[1] http://twitter.com/
[2] https://www.facebook.com/srch.php

tion estimation can be effective for this type of data.

To better understand our data, we analyze the properties of the queries as well as the associated field distribution information. In sum, we find that (1.) the queries collected using the MemRecap game could potentially be more realistic. (2.) Further analysis of field importance indicates that certain fields, such as the name of the poster, can be particularly important for retrieval. (3.) There is a high correlation between manual-specified query fields and fields estimated using field distribution models, which can be the key to better retrieval performance.

In the rest of the paper, we first introduce related work in Section 2 and detail the approach to building test collections in Section 3. The retrieval models studied are described in Section 4. Sections 5 and 6 respectively analyze the query set and the field information. In section 7, we show retrieval performance using different approaches, and conclude the paper in Section 8.

2. RELATED WORK

The task of searching personal social media collections is related to a number of research areas including building test collections, semi-structured document retrieval, known-item search, desktop search, and microblog search.

The difficulty of evaluating search on personal social data results from the lack of available test collections. Human computation games [8, 7] suggest a way of collecting queries based on an interaction that involves asking users to search in response to displayed documents. We extend the DocTrack game [7] by enabling users to manually specify field information. We further propose another type of game, MemRecap, as an attempt to obtain realistic queries in the absence of real query logs.

Social media data is usually described with a rich set of fields in the form of semi-structured documents. Previous work [6, 9, 12, 13] has explored modeling field importance for retrieval, as associating words with fields can be viewed as a hidden process in query formulation. Kim et al.[6] proposed a probabilistic retrieval model based on collection statistics. We explore several new ways to estimate field weights, as well as the feasibility of interpolating various language models based on a series of field distribution estimations.

Searching personal archives including email and documents on desktop machines has been studied to some extent [9, 18, 3, 4], and is usually modeled as a task of known-item search[5]. The accumulation of personal social data can be viewed as a new type of collection on the desktop; however, little is currently known about the best search techniques for this type of data. The emergence of new tools such as Greplin[3] has provided web services for personal search engines.

Our work is related to microblog search such as the official Twitter Search which globally retrieves documents (tweets) from the entire collection. However, the focus of Twitter Search is not on individuals or personal archives and documents may often be ranked based solely on time[2]. Other aspects of searching Twitter have been explored, including event detection [14][17], locating topics of interest[10][19], and mining opinions[16]. Facebook search emphasizes social relationships, and provides search based mainly on people or organizations. Our research, while using data collected from Twitter and Facebook, differs from the previous work in that we focus on searching personal archives.

[3]https://www.greplin.com/

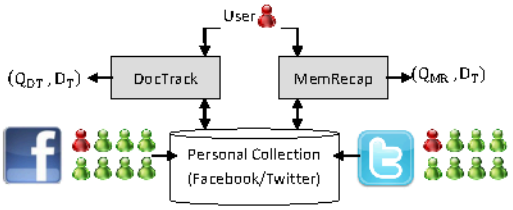

Figure 1: Overview of a personal social media collection.

Table 1: Statistics of Facebook and Twitter data.

Facebook			Twitter		
User	#Q	#Doc	User	#Q	#Doc
user0	35	19741	user5	41	127512
user1	67	165068	user6	58	24835
user2	36	47067	user7	50	92891
user3	62	28128	user8	62	14805
user4	60	42224	user9	60	32714

3. BUILDING PERSONAL SOCIAL MEDIA COLLECTIONS

In this section, we detail the steps taken to build personal social media collections, which include documents, queries, and relevance judgments. A personal document collection contains a set of documents that an individual could have seen previously by using social media applications. In practice, we assume the possible set of viewed documents for a user is composed of his own postings as well as all other postings of his friends from either Facebook or Twitter. For each personal document collection, we also collect two types of known-item queries, Q_{DT} and Q_{MR}, based on the human computation games DocTrack and MemRecap. Figure 1 provides a high level overview of a personal social media collection.

3.1 Document Collection – Facebook

To collect the set of documents that a user may have viewed, we obtained authorization to gather the postings related to a set of users and their friends within a period of time[4] using the Graph API [5] supported by Facebook. We store each posting as a XML file, which is regarded as a document in the collection. Table 1 shows the total number of documents collected for each of the 5 individuals involved in our experiments.

Facebook data is rich both in terms of the various types of postings as well as XML fields. The most common types are "status", "link", "photo" and "video". The XML fields contain nested information over a collection of sources. Given that our focus is on conducting known-item search on the social media collections, we flatten the field structure and keep only the more informative and interesting content. Specifically, we merge some of the fields and in total use 14 single-layered fields including uname (usernames of the posters and/or receivers), msg (main body of text in the post),

[4]07/01/2010-02/01/2011

[5]https://developers.facebook.com/docs/reference/api/

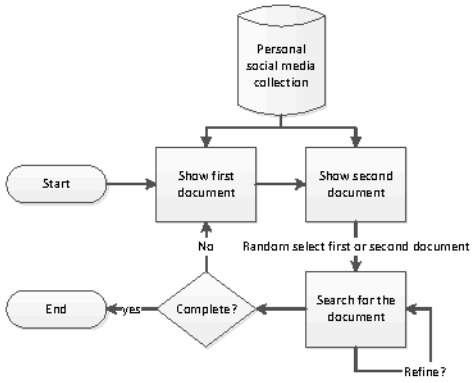

Figure 2: Flowchart of collecting Q_{DT} based on the DocTrack game.

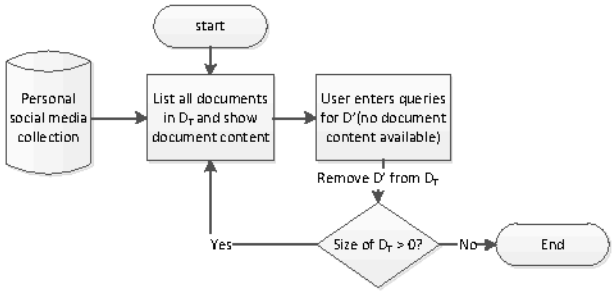

Figure 3: Flowchart of collecting Q_{MR} based on the MemRecap game.

pct (post created time), put (post updated time), cmtname (usernames involved in the comments), cmtmsg (all comment texts), ct (comments created time), likename (usernames who "like" the post), link (content of the post is typed as "link"), description (texts describing link content), caption (title of links), olink (other links that may appear any place in the post), pname (usernames involved in photos), and aname (usernames involved in albums).

3.2 Document Collection – Twitter

Similar to Facebook, we construct a personal document collection for Twitter by gathering postings of a set of individuals together with postings from the people they follow. We have collected 5 personal collections from Twitter. The statistics are shown in Table 1.

Again, we discard XML field information such as "the background color of a user profile" that is unlikely to be used in a query by users. In total, we keep 11 flattened single-layered fields consisting of text (main body of tweet text), uname (usernames of poster), replyname (usernames of people being replied), src (source of media used for tweeting such as smart phones or web), time (post created time), link (links that may appear any place in the post), re-text (main body of retweeted text), re-uname (usernames of poster being retweeted), re-replyname, re-src, and re-time.

3.3 Query Set – DocTrack

To complete the test collections for known-item search, test queries along with the corresponding relevance judgments are required. The task can be done with human computation games that reduce the cost of collecting data as well as providing higher motivation for users' participation. The DocTrack game proposed by Kim et al [7] functions by first showing two documents to users within a limited period of time, and asking them to randomly search one of the documents by typing in a query. Following [7], the search system was implemented using Apache Lucene[6] which differs from the retrieval models in Section 4 and avoids biased results towards certain models. Users can refine their queries for about 3 times if they cannot find the target documents. DocTrack simulates the situation that a user has seen a document before, and tries to recall it later. The randomness of choosing one of the two documents is designed to make it more difficult for users to formulate queries. Following this procedure, pairs of DocTrack query and target documents $\{(Q_{DT}, D_T)\}$ can be collected. For a known-item query, the relevance judgments are also provided by the association between each pair of (Q_{DT}, D_T). Figure 2 illustrates the flow chart of the DocTrack game.

Occasionally, users remember information such as "the person who wrote a message" or "the time when a posting was posted". We assume that this type of structural information could potentially help retrieve target documents. We therefore extended the DocTrack interface by allowing users to specify fields from which a query term comes during the process of typing queries. For example, given a query "Jessica basketball game", we provide an optional method for users to specify a structured query as "Jessica.user basketball.message game.message".

3.4 Query Set – MemRecap

Despite the effectiveness of DocTrack as demonstrated in previous research [7], the queries Q_{DT} can still be different from real queries that can only be gathered via a real search log system on personal collections. Moreover, we found some potential improvements that can be made for the DocTrack game. The time restriction on reading documents is different from a typical scenario of reading social media postings on the web. In addition, as DocTrack requires users to search the target document after reading, the search algorithm could potentially affect how users formulate queries. Also, we follow the original design of DocTrack game in [7] that all the query terms should be exactly matched with the terms in documents. That is, in order to retrieve the target documents, users need to formulate queries using words that are 100% from the target documents.

In an attempt to produce more realistic queries, we designed the second game, MemRecap, as illustrated in Figure 3. MemRecap takes the same set of target documents $\{D_T\}$ previously collected by DocTrack as input. During each round of MemRecap, users are asked to read as many $d \in \{D_T\}$ as they want without time limitation, under the constraint that users are confident of memorizing all of them. The set of documents read is denoted as D'. After users finish reading, they are directed to another page where the contents of the documents are no longer available and where they can type queries for D'. As users may have read quite a few documents in each round, we provide users with hints which help them correctly remember the documents. In particular, we show the profile pictures of people involved in each document to the users, which could strike a balance of

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

providing hints as well as keeping the words in the document hidden. Users can start another round of reading documents after they finish formulating queries, and the process repeats until there is no more $d \in \{D_T\}$ unspecified.

MemRecap gathers a new set of queries $\{Q_{MR}\}$ based on the target documents D_T previously collected by DocTrack. This makes us possible to compare the queries $\{Q_{DT}\}$ and $\{Q_{MR}\}$ that share the same source document. Moreover, MemRecap makes modifications based on DocTrack by removing the time constraint and the search requirement. Also, MemRecap provides a free-text interface for users to type their queries. In such case, users are not required to formulate queries using only the words from target documents, which can be realistic in some cases where users have vague memories.

For MemRecap, we do not collect field information as in the DocTrack game. As users can choose as many documents to read as they want, it could be difficult for them to accurately remember which field a query word should belong to, as opposed to the two documents in DocTrack. The average number of documents that users read in each around is four.

4. RETRIEVAL MODELS

Social media documents contain richer field information and fewer words than typical documents. In this section, we propose several retrieval methods that could potentially be suited to the search task. We denote $Q = \{q_1, q_2, \ldots, q_n\}$ as a query sample from either Q_{DT} or Q_{MR}, and use $f = \{f_1, f_2, \ldots, f_m\}$ to represent the fields of a document. Further, $F = \{F_1, F_2, \ldots, F_m\}$ denotes the fields from the perspective of collections; that is, each F_j is composed of f_j across all the documents in the collections.

4.1 Query likelihood model (QLM)

We use the well-known query likelihood model as a baseline for retrieval, which ranks documents according to the probability of generating a term q_i from a document d.

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

4.2 Proximity model (Prox)

A proximity model helps identify phrases by emphasizing words appearing within a pre-defined window[7]. The Prox model helps to test the assumption that query terms are likely to appear in close proximity to one another in relevant documents, and is formulated as:

$$P_{Prox}(Q|d) = P_{unordered}(q_1 \ldots q_k|d)$$

4.3 Field specific model (FSM)

The correct association of a query term with its source field could potentially help retrieval performance. Given information about the association (q_i, f_j), we can retrieve documents by taking into account only the appearance of query term q_i in a certain field f_j as:

$$P_{FSM}(Q|d) = \prod_{i=1}^{n} P_{QL}(q_i|f_j, d)$$

[7]The window size is 8 in our experiments.

Specifically, the probability $P_{QL}(q_i|f_j, d)$ is calculated by dividing the number of times q_i appearing in f_j by the length of entire document. In our experiments, FSM is only used for the Q_{DT} queries as we collect field information using the DocTrack game.

4.4 Field distribution model (FDM)

Since the correct association of words and fields is not always available, an alternative that estimates field weights for semi-structured documents has been shown to be effective in previous research [6, 9, 12, 13]. In this approach, a set of weights $w = \{w_1, w_2, \ldots, w_m\}$ over a set of fields $f = \{f_1, f_2, \ldots, f_m\}$ is estimated for each of query term q_i; that is, a probability distribution over the fields is estimated for each q_i. Mathematically, the family of FMD models can be formulated as:

$$P_{FDM}(Q|d) = \prod_{i=1}^{n} \sum_{j=1}^{m} w_{ij} P_{QL}(q_i|f_j, d)$$

In the following sections, we describe an existing weight estimation technique as well as several proposed estimators. Alternatives exist for the calculation of $P_{QL}(q_i|f_j, d)$: to divide the number of times q_i in f_j by the length of field f_j or by the length of entire document d. Finally, we combine together all of the FDM models.

4.4.1 *Probabilistic Retrieval Model for Semi-structured Data* (PRMS)

The PRMS model proposed by Kim et al [6] estimates field importance based on statistical information from document collections. Kim et al [6] reported improvements in effectiveness for PRMS compared to several other field-based retrieval methods[12, 13]. The PRMS model computes a weight as follows:

$$w_{ij} = P(F_j|q_i) = \frac{P_M(q_i|F_j)P_M(F_j)}{\sum_{F_j \in F} P_M(q_i|F_j)P_M(F_j)}$$

Specifically, the weight w_{ij} corresponds to a posterior probability $P(F_j|q_i)$ that calculates the probability of mapping a query term q_i to a field F_j. The PRMS model focuses on the term distribution in each field by considering the probability $P_M(q_i|F_j)$ that measures how many times a q_i is included in a F_j. $P_M(F_j)$ is a prior representing how much belief one has in the field F_j, and is uniformly set according to Kim et al.

PRMS calculates the $P_{QL}(q_i|f_j, d)$ by division by the length of field f_j. For comparison, we add a variation of PRMS using the division by the entire document length, and denote this variation as FDM_{prms}. Except for PRMS, we adopt the definition of division by the entire document length thorough the paper.

4.4.2 *Probabilistic Model using All Fields* (FDM_{AllF})

In contrast to PRMS, we can also use the term distribution over all fields, with the advantage of estimating field importance according to the entire collection. In particular, we calculate the probability, $P(q_i \in F_j|q_i \in F)$, that measures how likely a term q_i belongs to a field F_j compared to all other fields F in the collection. $P(q_i \in F_j|q_i \in F)$ can be efficiently calculated based on the number of times q_i appears in each F_j. w_{ij} can be computed accordingly by normalizing over all possible fields for a q_i.

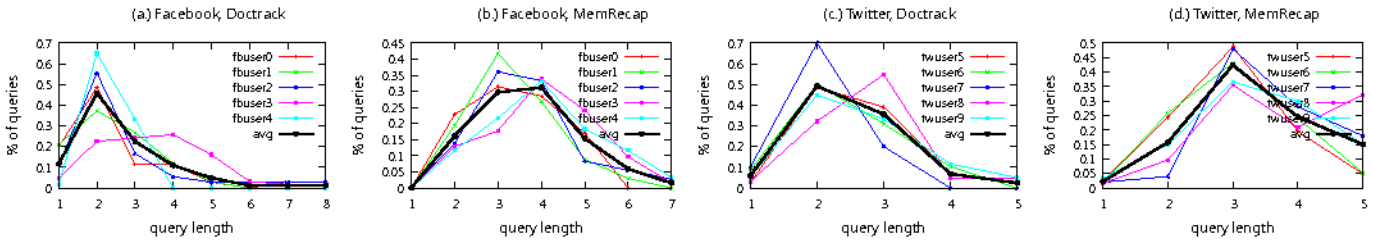

Figure 4: Query length distribution for DocTrack and MemRecap queries on Facebook and Twitter.

$$w_{ij} = \frac{P(q_i \in F_j | q_i \in F)}{\sum_{F_j \in F} P(q_i \in F_j | q_i \in F)}$$

4.4.3 *Probabilistic Model using Field Size* ($\text{FDM}_{\text{Fsize}}$)

We further study the feasibility of using the size of fields as an indicator for weight estimation. We assume that the more words a field contains, the more likely it is that a query term belongs to that field. We also normalize the weight by Z:

$$w_{ij} = \frac{1}{Z} \frac{|F_j|}{\sum_{F_j \in F} |F_j|}, \quad Z = \sum_{F_j \in F} \frac{|F_j|}{\sum_{F_j \in F} |F_j|}$$

4.4.4 *Probabilistic Model on Field Prior* ($\text{FDM}_{\text{Fprior}}$)

Different characteristics of the fields make them not equally important for search. We can incorporate prior knowledge concerning the effectiveness of different fields in the retrieval process. The $Prior(F_j)$ can be viewed a function for estimating how much belief one has for a certain field, and can be estimated in various ways. In Section 6.2, we show the estimation of $Prior(F_j)$ based directly on retrieval effectiveness.

$$w_{ij} = \frac{Prior(F_j)}{\sum_{F_j \in F} Prior(F_j)}$$

4.4.5 *Mixture of Field distribution model* (FDM_{mix})

Correct field importance estimation can be challenging. To this end, we incorporate together all of the FDM with each as a basic indicator, where the aim is to potentially combine the advantages and reduce the disadvantages of each of the models. We linearly combine the confidence over the fields by specifying a set of parameters $\Gamma = \{\gamma_k\}$ to each FDM. Mathematically, w_{ij} is estimated as,

$$w_{ij} = \frac{1}{Z} \sum_k \gamma_k w_{ij}^k, \quad Z = \sum_{F_j \in F} \sum_k \gamma_k w_{ij}^k$$

where w_{ij}^k corresponds to the estimation of the k^{th} FDM. In total, we use the models FDM_{prms}, FDM_{AllF}, $\text{FDM}_{\text{Fsize}}$, and $\text{FDM}_{\text{Fprior}}$, and $\Gamma = \{\gamma_k\}$ are uniformly distributed without prior knowledge.

4.5 Mixture of language models

A linear interpolation of individual retrieval models helps estimate the overall joint conditional probability based on several conditional probabilities. Accordingly, we combine QLM, field-related models(FSM/FDM) and Prox models using either two (Mix_{LM2}) or three (Mix_{LM3}) of them.

Mix_{LM2} linearly combines QLM with FSM, Prox, or FDM_{mix} models, which rank documents in proportion to the interpolated probabilities. $\lambda \in [0, 1]$ is a free parameter specifying the importance for each model, and is cross-validated and set to 0.8 in this paper, which is consistent with [1].

QLM+FSM: $rank(d) \propto \lambda P(Q|d) + (1-\lambda) P_{fsm}(Q|d)$
QLM+Prox: $rank(d) \propto \lambda P(Q|d) + (1-\lambda) P_{prox}(Q|d)$
QLM+FDM_{mix}: $rank(d) \propto \lambda P(Q|d) + (1-\lambda) P_{Fmix}(Q|d)$

Mix_{LM3} further combines 3 types of language models together. λ_Q, λ_F and λ_P are parameters controlling the weights of different models, with the sum of them equal to 1. λ_Q, λ_F and λ_P are empirically set to 0.8, 0.1 and 0.1, which is consistent with [11] in a sense that QLM is most highly weighted.

QLM+FSM+Prox:
$rank(d) \propto \lambda_Q P(Q|d) + \lambda_F P_{fsm}(Q|d) + \lambda_P P_{prox}(Q|d)$
QLM+FDM_{mix}+Prox:
$rank(d) \propto \lambda_Q P(Q|d) + \lambda_F P_{Fmix}(Q|d) + \lambda_P P_{prox}(Q|d)$

5. ANALYSIS ON DOCTRACK AND MEMRECAP QUERY SETS

In this section, we compare the two types of test queries, Q_{DT} and Q_{MR}, from statistical and retrieval effectiveness perspectives. Our finding suggests that the Q_{MR} queries can potentially be regarded as more similar to real queries than Q_{DT} queries in our environment.

5.1 Query length distribution

We first examine the distributions of query lengths for the DocTrack and MemRecap queries. Figure 4(a)(b) show the percentage of queries of different lengths for Q_{DT} and Q_{MR} in Facebook collections. It can be observed that there is around 10% of Q_{DT} queries with length 1, compared to 0% in Q_{MR}. When users are formulating Q_{DT} queries, it is more likely for them to randomly pick a word to remember as a result of time limitations on reading documents. In this case, the intention of users, according to the design of the DocTrack game, is to find the target document rather than to recall something that they have seen in the past. Another reason for the one-word queries in DocTrack is because users tend to refine the query by downsizing and dropping query words if they could not retrieve the target documents; this may imply the querying behavior is influenced by the search system in DocTrack.

For Q_{MR}, users formulate their queries using at least two words. The time restriction of DocTrack also makes the

Table 2: Average t-test p-values between MRR samples of DocTrack and MemRecap queries. Results are reported for Facebook and Twitter collections using various retrieval models.

p-value	QLM	Prox	PRMS	F_{prms}	F_{AllF}	F_{Fsize}	F_{Fprior}	F_{mix}	Q+P	Q+F_m	Q+F_m+P
Facebook avg	0.9989	0.0018	0.7658	0.7836	0.9989	0.9989	0.9989	0.9807	0.9989	0.9989	0.9989
Twitter avg	0.9766	0.0001	0.7612	0.7712	0.9677	0.9677	0.9677	0.9612	0.9767	0.9877	0.9877

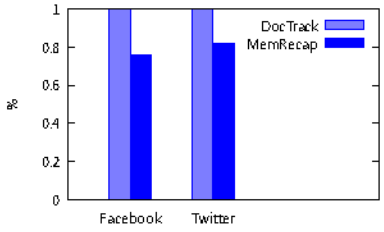

Figure 5: Average term overlap rate for DocTrack and MemRecap queries on Facebook and Twitter.

query length of Q_{DT} (average 2.68) shorter than Q_{MR} (average 3.69). Similarly, we find that 45% of DocTrack queries have query length 2, while query lengths of 3 (29%) and 4 (31%) are more common in MemRecap queries. For example, "pic" and "pic being tan" are the DocTrack and MemRecap queries for the same target document.

Moreover, a small portion of DocTrack queries (3% averagely) were very long because users could randomly memorize a sentence under the pressure of limited time, which can be unrealistic. For example, "not getting enough sleep is bad" and "lecture sleeping" are the DocTrack and MemRecap queries for the same target document. For the case of MemRecap queries, long queries of length 6 or 7 are mainly composed of "uname" together with several other words describing the content of "msg". One example of long queries in Q_{MR} is "bon jovi suffer olympic stadium finland".

Similarly, Figures 4(c)(d) show the distribution of query lengths for Q_{DT} and Q_{MR} in the Twitter collections. The length of the majority of Q_{MR} queries is one word longer than Q_{DT} queries. It can also be observed that, comparing queries across Facebook and Twitter, queries for Facebook are usually longer as a result of postings having more information in Facebook archives.

5.2 Term overlap rate

We examined the rate of exact matches between query and document words in Figure 5. According to the design of DocTrack game, there is 100% term overlap rate in all cases, while there is about 20% of query words in Q_{MR} that are not included in D_T. We believe both types of queries can be useful for testing the effectiveness of our retrieval algorithms. Q_{DT} can represent the case that users have sharp memories about the known documents, whereas Q_{MR} provides some degree of uncertainty to represent the case where users could have vague memories. Also, we examine the non-matched words of Q_{MR} and find that reasons such as misspelling, different verb tenses, synonyms, or simply errors can account for much of the 20% mismatch.

From Figure 5, we can see that the term overlap rate of Q_{MR} is higher for Twitter collections. This is because the length of Twitter documents are usually shorter which makes it easier for users to recap the document content.

Table 3: Statistics of field information for Q_{DT}.

Facebook			Twitter		
user	#words	Perc%	user	#words	Perc%
0	78	0.9487	5	106	0.9528
1	160	0.9937	6	141	1.0000
2	78	0.5512	7	105	0.9904
3	217	0.9493	8	171	1.0000
4	139	0.9064	9	124	0.9012
avg	134.4	0.8699	avg	129.4	0.9686

5.3 Retrieval performance deviation

Comparison of the retrieved results is useful for understanding the degree of compatibility between queries from the perspective of search systems. Accordingly, we compare the distribution of retrieval scores between DocTrack and MemRecap queries. Specifically, the two-tailed student's t-test, which has the advantage of being suitable for small datasets and being sensitive if the data meets the requirements of the test, determines if two datasets differ significantly based on the null hypothesis. Based on this test, we conclude that two distributions are not significantly different from each other if the resulting p-value of the two empirical samples is greater than a certain threshold.

Table 2[8] reports the average p-values over 5 users between Q_{DT} and Q_{MR} respectively for both Facebook and Twitter collections. From Table 2, the two types of queries consistently agree with each other, except for the Prox model. The significant difference (p-value < 0.05) for the Prox model suggests that users tend to memorize consecutive words from target documents in order to generate queries in DocTrack games. However, in the MemRecap game, users do not exhibit the same tendency as they were given sufficient time to go through the content of the documents in detail. For the other retrieval models, Q_{DT} and Q_{MR} are similar to each other.

6. ANALYSIS ON FIELD INFORMATION

Another focus of this paper is to analyze the field distribution of query words and how this distribution can potentially affect retrieval performance. In this section, we describe the distributions for the manual-specified fields as well as the estimated fields based on the retrieval model FDM_{mix}. Note that we collect manual-specified field information only based on the DocTrack game, and therefore we focus on the Q_{DT} queries here.

6.1 Statistics of Field Information

Though not required, users are recommended to specify fields for query words in DocTrack game. In Table 3, we show the total number of words and the percentage of words that are specified with fields. We can see that users are generally willing to specify fields when formulating queries

[8]Abbreviations: Q(QLM), P(Prox), F_m(FDM_{mix}).

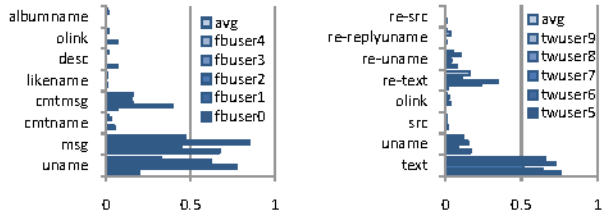

Figure 6: Field distribution for manual-specified queries for Facebook (left) nd Twitter (right).

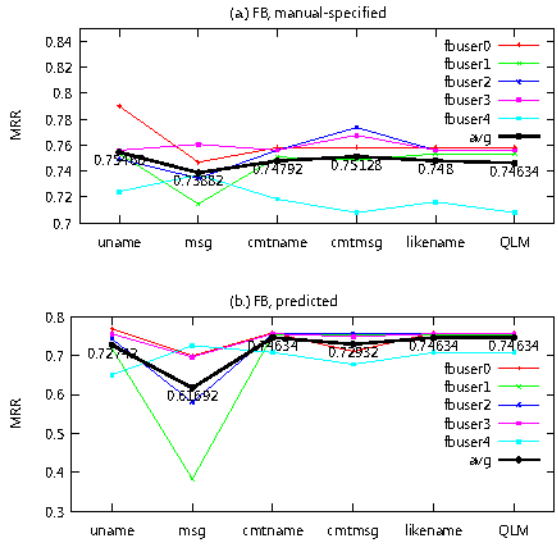

Figure 7: MRR of QLM and FSM using human-specified/predicted field information on Facebook

(more than 85%), which makes the following analysis more reliable.

Moreover, for the set of query words manually specified with fields, we are interested in how the field information is distributed. From Figure 6 [9], it can be observed that most query words are associated with the "msg"/"text" (46% in Facebook and 65% in Twitter) or "uname" (32% in Facebook and 11% Twitter) fields. Words specified with "cmtmsg" (15%) or "re-text" (15%) also share a fair portion of all the fields, while the rest of fields are seldom addressed. The fields involving time information are never specified.

6.2 Field importance for retrieval

A straightforward approach to understanding the importance of fields for retrieval is to search using each of the fields individually. As a baseline for comparison, we first retrieve documents using Q_{DT} based on QLM where fields do not get involved. Then for each field F_j, we search with a FSM model where words specified with F_j are restricted to that field while others remain the same as in QLM. For example, "Jessica basketball.message game.message" would be the query we issue if we were to inspect the performance impact of field "message". Figure 7(a) and Figure 8(a) show the MRR results of search using the F_j-field-specific FSM

[9]Fields that are never specified are omitted in the figures

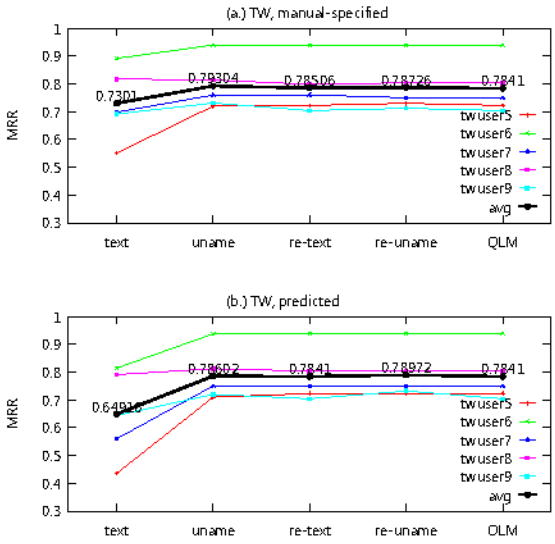

Figure 8: MRR of QLM and FSM using human-specified/predicted field information on Twitter.

model as well as the QLM model[10].

We also explore the impact of fields when the manual-specified field information is no longer available. In this case, for each query term q_i, as an approximation, we adopt the top-weighted field $F_{top1} = \arg\max_{w_j} F_j$ as the field which q_i belongs to. Specifically, we use the FDM_{mix} model to estimate field weights and select F_{top1} for each q_i. Similarly, we conduct search based on field-specific FSM model for each F_j, and the retrieval performance can be found in Figure 7(b) and Figure 8(b). From these figures, we summarize several findings:

- Comparing manual-specified and predicted fields, the retrieval performance using predicted information is worse. This indicates that using only a single field suffers from incorrect prediction and inevitably decreases performance.
- The agreement rates between manual-specified and predicted fields are 64% and 70% for Facebook and Twitter collections.
- For facebook data, "uname" is more helpful than other fields both with (Fig. 7(a)) or without (Fig. 7(b)) manual specifications. This supports the intuition that named entities such as person names show strong correlation with better performance, and that an interface design that emphasizes specifying person names in the query could potentially help retrieve relevant documents. The specification of "msg" field, contrary to our expectation, is no better than simple QLM. This may be because that the words in message fields are more common and can be found in several fields, and therefore restricting to the message field does not provide further information.
- For Twitter data, similar trends, such as better effectiveness from "uname" can be found. In particular,

[10]The default Dirichlet smoothing with mu=2500 is used.

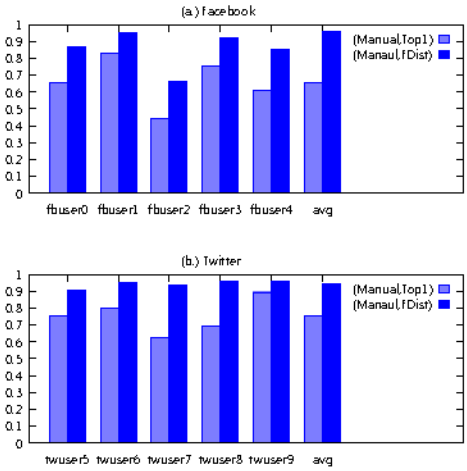

Figure 9: Correlation of (V_{avg}^{man}, V_{avg}^{top1}) and (V_{avg}^{man}, V_{avg}^{FDM}) on Facebook and Twitter.

the restriction to "text" results in a large drop of performance compared to QLM. This could be the result of frequent retweets in the Twitter corpus. For the case of retweets, QLM considers a query term q_i twice in "text" and "re-text"while FSM counts just once for either of the 2 fields.

Finally, let us recall the $\text{FDM}_{\text{Fprior}}$ model in Section 4. In this paper, we normalize the retrieval performance across the fields to estimate the function $Prior(F_j)$. Specifically, we take 20% of queries from each user for $Prior(F_j)$ estimation.

6.3 Field distribution of query terms

In addition to associating q_i with only a single field as in Section 6.2, we now focus on understanding the field distribution for each query term. First we introduce the notations: vector $V_i^{man} = (v_1, v_2, \ldots, v_m)$ describes the manual-specified field distribution for a query term q_i. For example, $(1, 0, \ldots, 0)$ can be used to describe q_i is manually specified "uname" in Facebook. If q_i is not manually specified with a field, V_i^{man} is then uniformly distributed. Vector V_i^{top1} on the other hand represents the estimated field distribution for q_i, and is always composed of a series of "0" or "1" as only the top-weighted field is considered. Finally, the vector V_i^{FDM} records the real-valued field distribution for each q_i based on the estimation of FDM_{mix} model.

For each query term q_i, we may describe it using the three vectors V_i^{man}, V_i^{top1}, and V_i^{FDM}. We are then able to summarize the characteristic of each personal collection with 3 average vectors V_{avg}^{man}, V_{avg}^{top1}, and V_{avg}^{FDM} by taking the average over the set of query terms. Moreover, we compute the Pearson product-moment correlation coefficient of (V_{avg}^{man}, V_{avg}^{top1}) and (V_{avg}^{man}, V_{avg}^{FDM}) for each personal collection, as shown in Figure 9. The two figures show a consistently higher correlation of (V_{avg}^{man}, V_{avg}^{FDM}) than (V_{avg}^{man}, V_{avg}^{top1}). This finding shows that modeling q_i using a field distribution is reasonable. Also, in the following section, we show the retrieval performance based on field distributions can be very effective and even better than manual specifications.

7. RETRIEVAL EXPERIMENTS

In this section, we compare the retrieval effectiveness of the various models described previously. Firstly, we describe the experimental setup. Then, we detail the retrieval performance of the models for the Facebook and Twitter collections.

7.1 Experimental Setup

The test collections used for evaluation include 10 personal social media collections, with 5 gathered from Facebook and the other 5 from Twitter. For each personal collection, the query set is composed of two types of queries, Q_{DT} and Q_{MR}, which are collected based on DocTrack and MemRecap respectively. To ensure the quality of queries, we keep queries that have their target documents ranked in the top 50 by the DocTrack game as suggested in Kim et al, which may cause different numbers of queries for each user. Table 1 details the statistics of the collections. Moreover, we use the mean reciprocal rank (MRR) for effectiveness measurement as we focus on the task of known-item search. There is only one relevant document for each query, and the relevance judgments are the same for the Q_{DT} and Q_{MR} queries.

We index the personal collections using the Indri toolkit[11], where documents are stemmed with the Krovetz stemmer and no stopwords are eliminated. We show the effectiveness across various retrieval models discussed in Section 4, including QLM, FSM, FDM with different estimators, and finally the mixture of the language models. Specifically, we cross-validated and fixed the smoothing parameters for each retrieval model.

7.2 Performance on Facebook Collections

In the following, we summarize several observations from Table 4 and Table 5. Table 4[12] shows the retrieval performance for Q_{DT} on the Facebook collections. (1.) We see that the simple QLM model (the baseline model) can be very effective with an average MRR of 0.7519. FSM, though with extra field information, is merely comparable to QLM as field specification can sometimes be too restrictive. The Proximity model alone is not as effective, indicating that query words can sometimes be scattered in the documents. (2.) For the FDM models, including PRMS, FDM_{PRMS}, FDM_{AllF}, $\text{FDM}_{\text{Fsize}}$, and $\text{FDM}_{\text{Fprior}}$, have similar results and are close to the performance of QLM. We further take a uniform mixture of FDM models to form FDM_{mix}, whose performance is the second best of all FDM models. Though FDM_{mix} is not the top-performing FDM model, the effect of combining different FDM estimations can be complementary, and reduces the risk of being biased to current collections.

From Table 4, (3.) we see that combining different language models including Mix_{LM2} and Mix_{LM3} can significantly improve retrieval effectiveness, with Mix_{LM3} outperforming Mix_{LM2}. This shows that combining evidence from the document, the fields, and the words location is beneficial to retrieval. (4.) The use of manual-specified fields can be more precise than field distribution estimated by FDM_{mix} for the combined models, and thus QLM+FSM and QLM+FSM+Prox are consistently better than QLM+FDM_{mix} and QLM+FDM_{mix}+Prox.

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

[12] Abbreviations:Q(QLM), F(FSM), P(Prox), F_m(FDM_{mix}).

Table 4: MRR of Q_{DT} on Facebook collections. * is marked for p-value < 0.05 compared to QLM.

user	QLM	FSM	Prox	PRMS	F_{prms}	F_{AllF}	F_{Fsize}	F_{Fprior}	F_{mix}	Q+F	Q+P	$Q+F_m$	Q+F+P	$Q+F_m$+P
0	0.7581	0.7833	0.7254	0.7643	0.7842	0.7603	0.7825*	0.7851*	0.7892*	0.7963*	0.7967*	0.7798	0.8102*	0.8110*
1	0.7643	0.7012*	0.5360*	0.6474*	0.7225*	0.6791*	0.7218	0.7333	0.7057*	0.7846	0.7723	0.7617	0.7891	0.7663
2	0.7608	0.7743	0.4844*	0.7397	0.7504	0.7638	0.7589	0.7706	0.7799	0.7654	0.7513	0.7951	0.7875	0.7851
3	0.7712	0.8142*	0.7585	0.8351*	0.7377*	0.7582	0.7636	0.7716	0.7634	0.8046*	0.8328*	0.7623	0.8430*	0.8337*
4	0.7051	0.7114	0.5201*	0.7037	0.7026	0.7190	0.7190	0.7383*	0.7422	0.7829*	0.7398	0.7134	0.7735*	0.7439*
Avg	0.7519	0.7568	0.6048*	0.7380	0.7394	0.7360	0.7491	0.7597	0.7560	0.7867	0.7785	0.7624	0.8006*	0.7880*

Table 5: MRR of Q_{MR} on Facebook collections. * is marked for p-value < 0.05 compared to QLM.

user	QLM	Prox	PRMS	F_{prms}	F_{AllF}	F_{Fsize}	F_{Fprior}	F_{mix}	Q+P	$Q+F_m$	$Q+F_m$+P
0	0.7781	0.5101*	0.7631	0.7558	0.7822	0.7930	0.8009	0.7916	0.8000	0.8167*	0.8236*
1	0.7824	0.4225*	0.6928*	0.7472	0.7425*	0.7756	0.7756	0.7493	0.7725	0.7929	0.7834
2	0.7447	0.2092*	0.7460	0.7399	0.7216	0.7137	0.7430	0.7268	0.7446	0.7447	0.7582
3	0.7995	0.5364*	0.7978	0.8129	0.8303*	0.8212	0.8214*	0.8231*	0.8573*	0.8044	0.8745*
4	0.7239	0.3164*	0.6859	0.7122	0.6753*	0.7092	0.7071	0.7132	0.7417	0.7415	0.7533
Avg	0.7657	0.3989*	0.7371	0.7536	0.7503	0.7625	0.7696	0.7608	0.7832	0.7800	0.7986*

For the other test query set Q_{MR}, we report the MRR in Table 5 for each retrieval model except for the ones associated with manual-specified fields (e.g., FSM, QLM+FSM and QLM+FSM+Prox). From Table 5, (5.) similar trends such as that the basic QLM sets a high baseline and Mix_{LM3} outperforms all other approaches can be observed. There are, however, differences compared to Q_{DT}. (6.) The Prox model works significantly worse for Q_{MR}, which we believe is because users have more time on reading in MemRecap and formulate queries based on the entire document. In this case, the query words of Q_{MR} can be more scattered than Q_{DT}, causing poor effectiveness for the Prox model alone.

Finally, (7.) comparing performance across Q_{DT} and Q_{MR}, we can see that queries collected based on MemRecap tend to be more effective despite the fact that words in Q_{DT} are exactly matched with the target documents. Several reasons have been discovered for this. Users have more time to understand document contents in MemRecap. Also, with a search-based mechanism, users tend to downsize the query length in the DocTrack game if they have incorrectly memorized document content and couldn't locate the target document. It seems more natural for users to formulate queries using stemmed (root) vocabularies, as can be observed in MemRecap queries. Considering the exact match approach in DocTrack, non-stemmed vocabularies such as past tense verbs could cause word mismatch between queries and documents.

7.3 Performance on Twitter Collections

We also show the retrieval performance of Q_{DT} and Q_{MR} for Twitter personal collections, respectively in Table 6 and Table 7. The overall trend of performance is consistent with the results of Facebook collections. The QLM model sets a high baseline for finding the target documents. The individual FDM models do not outperform the QLM model; however, together Mix_{LM2} and Mix_{LM3} can significantly improve the baseline performance. The Prox model alone works poorly for MemRecap queries as previously.

Comparing performance across the Facebook and Twitter collections, we can see that the effectiveness of retrieving Twitter documents is on average higher than that of Facebook collections. In this case, the shorter length of Twitter documents makes it easier for retrieval models to identify the target documents as there is less information contained in the tweets. For the same reason, we find it is challenging to improve baseline performance using the mixture of language models. The limitation comes from the fact that exploiting field structures as well as proximity information can still be similar to just using the plain text of document, as the document can be too short to contain extra text information.

8. CONCLUSION

In this paper, we propose the viewpoint that social media archives can serve as a new type of personal documents in the desktop environment. It is clear that providing search tools can be important; however, the lack of publicly available test collections makes evaluation difficult. To address this, we describe how text collections for personal social data can be created. In particular, we use human computation games, Doctrack and MemRecap, to collect known-item queries from real users. We also explore a range of retrieval models that exploit the semi-structured nature of social data.

Our results have several interesting aspects. The retrieval experiments show that a mixture of language models Mix_{LM3} with field distribution estimation can be effective for this type of data. Also, we see that certain fields, such as poster name, can be particularly important for retrieval. This provides a hint for future search applications that letting users specify names can potentially enhance retrieval performance.

This work has focused on evaluating search using a single collection from either Facebook or Twitter. In future work, it would be important to deal with multiple types of social media documents simultaneously, which can involve the task of type prediction as in the desktop environment. In addition, the nested XML structures could potentially contain information that could improve retrieval, but is currently not being considered in this paper. Finally, we are implementing a personal search application on desktop and mobile platforms. With this search tool, not only can we satisfy the user needs of retrieving known items, but real queries can be acquired for further research.

Table 6: MRR of Q_{DT} on Twitter collections. * is marked for p-value < 0.05 compared to QLM.

user	QLM	FSM	Prox	PRMS	F_{prms}	F_{AllF}	F_{Fsize}	F_{Fprior}	F_{mix}	Q+F	Q+P	$Q+F_m$	Q+F+P	$Q+F_m$+P
5	0.7305	0.5841*	0.5728*	0.6831	0.7154	0.7215	0.6078*	0.7125	0.7051	0.7431	0.7516	0.7488	0.7541	0.7338
6	0.9402	0.9103	0.8678*	0.9389	0.9329	0.9326	0.9114	0.9431	0.9431	0.9454	0.9402	0.9431	0.9454	0.9402
7	0.7753	0.7148*	0.7319	0.7761	0.7624	0.7124*	0.6965*	0.7550	0.7320	0.7751	0.7702	0.7719	0.7584	0.7660
8	0.8353	0.8245	0.6855*	0.8002	0.8568	0.8264	0.7848	0.8156	0.8197	0.8393	0.8237	0.8286	0.8393	0.8259
9	0.7133	0.7012	0.6027*	0.7059	0.6984	0.7399	0.7053	0.7098	0.7286	0.7401	0.7339	0.7356	0.7531*	0.7459
Avg	0.7989	0.7469	0.6921*	0.7808	0.7931	0.7865	0.7411	0.7872	0.7857	0.8086	0.8039	0.8056	0.8100*	0.8023

Table 7: MRR of Q_{MR} on Twitter collections. * is marked for p-value < 0.05 compared to QLM.

user	QLM	Prox	PRMS	F_{prms}	F_{AllF}	F_{Fsize}	F_{Fprior}	F_{mix}	Q+P	$Q+F_m$	$Q+F_m$+P
5	0.7802	0.5102*	0.7396	0.7496	0.7419	0.7353	0.7310*	0.7401	0.7966	0.8056	0.8086
6	0.9064	0.6810*	0.8873	0.8943	0.8928	0.8814	0.9063	0.9031	0.9043	0.9064	0.9043
7	0.8443	0.5975*	0.8498	0.8203	0.7979*	0.8464	0.8484	0.8245	0.8475	0.8420	0.8475
8	0.9395	0.5376*	0.9307	0.9276	0.9260	0.9327	0.9327	0.9327	0.9395	0.9395	0.9395
9	0.7361	0.5668*	0.7204	0.7313	0.7200	0.7361	0.7457	0.7413	0.7638*	0.7519	0.7710*
Avg	0.8413	0.5786*	0.8255	0.8246	0.8157	0.8263	0.8328	0.8283	0.8503	0.8490	0.8541

9. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by ARRA NSF IIS-9014442. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

10. REFERENCES

[1] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR '08*, pages 491–498, 2008.

[2] C. Chen, F. Li, B. C. Ooi, and S. Wu. Ti: an efficient indexing mechanism for real-time search on tweets. In *SIGMOD '11*, pages 649–660, New York, NY, USA, 2011. ACM.

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

[4] 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.

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

[6] J. Kim and W. B. Croft. Retrieval experiments using pseudo-desktop collections. In *CIKM '09*, pages 1297–1306, 2009.

[7] J. Kim and W. B. Croft. Ranking using multiple document types in desktop search. In *SIGIR '10*, pages 50–57, New York, NY, USA, 2010. ACM.

[8] H. Ma, R. Chandrasekar, C. Quirk, and A. Gupta. Improving search engines using human computation games. In *CIKM '09*, pages 275–284, New York, NY, USA, 2009. ACM.

[9] C. Macdonald and I. Ounis. Combining fields in known-item email search. In *SIGIR '06*, pages 675–676, New York, NY, USA, 2006. ACM.

[10] K. Massoudi, M. Tsagkias, M. de Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR'11*, pages 362–367, Berlin, Heidelberg, 2011. Springer-Verlag.

[11] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *SIGIR '05*, pages 472–479, New York, NY, USA, 2005. ACM.

[12] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *SIGIR '03*, pages 143–150, New York, NY, USA, 2003. ACM.

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

[14] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW '10*, pages 851–860, New York, NY, USA, 2010. ACM.

[15] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: a comparison of microblog search and web search. In *WSDM '11*, pages 35–44, New York, NY, USA, 2011. ACM.

[16] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. *Predicting elections with Twitter: What 140 characters reveal about political sentiment*, pages 178–185. 2010.

[17] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Human factors in computing systems*, CHI '10, pages 1079–1088, New York, NY, USA, 2010. ACM.

[18] S. Yahyaei and C. Monz. Applying maximum entropy to known-item email retrieval. In *ECIR'08*, pages 406–413, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] X. Zhou, H. Chen, Q. Jin, and J. Yong. Generating associative ripples of relevant information from a variety of data streams by throwing a heuristic stone. In *ICUIMC'11*, pages 59:1–59:7, New York, NY, USA, 2011. ACM.