A Fixed-Point Method for Weighting Terms in Verbose Informational Queries

Jiaul H. Paik
UMIACS
University of Maryland
College Park, MD 20742 USA
jiaul@umd.edu

Douglas W. Oard
iSchool and UMIACS
University of Maryland
College Park, MD 20742 USA
oard@umd.edu

ABSTRACT

The term weighting and document ranking functions used with informational queries are typically optimized for cases in which queries are short and documents are long. It is reasonable to assume that the presence of a term in a short query reflects some aspect of the topic that is important to the user, and thus rewarding documents that contain the greatest number of distinct query terms is a useful heuristic. Verbose informational queries, such as those that result from cut-and-paste of example text, or that might result from informal spoken interaction, pose a different challenge in which many extraneous (and thus potentially misleading) terms may be present in the query. Modest improvements have been reported from applying supervised methods to learn which terms in a verbose query deserve the greatest emphasis. This paper proposes a novel unsupervised method for weighting terms in verbose informational queries that relies instead on iteratively estimating which terms are most central to the query. The key idea is to use an initial set of retrieval results to define a recursion on the term weight vector that converges to a fixed point representing the vector that optimally describes the initial result set. Experiments with several TREC news and Web test collections indicate that the proposed method often statistically significantly outperforms state of the art supervised methods.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval: Retrieval Models

General Terms

Algorithm, Experimentation, Performance

Keywords

Document ranking, Retrieval model, Term weighting

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661957.

1. INTRODUCTION

Users of widely used search engines typically interact with the system through free form natural language queries to address a broad range of information needs. Simple ranked retrieval techniques based on matching terms in the query and the document depend for their quality on the user entering the right set of keywords. When a user has a precise information need, this often works well. On some occasions, however, the user may not have a clear view of the potentially relevant documents they are looking for, and might choose to err on the side of entering more rather than fewer terms, thus creating a query that contains extraneous terms that may indeed mislead the search engine. Another common source for long queries is cut and paste, in which a user finds some text on some topic (e.g., a review of a new mobile phone) and simply drops that text the the query window (perhaps in the hope of finding more reviews?). A third scenario that might result in long queries could be a hands-free and eyes-free spoken search scenario (e.g., while driving a car) in which, in an effort to overcome the effect of speech recognition errors, the user might simply ramble on a bit, repetitively, in the hope that the system will eventually catch on and provide them with what they want.

Ultimately, we want search engines that work well with short queries when that's what they receive, and also with longer queries when that's what the user chooses to provide. Most well known retrieval models (BM25 [27], DFR [2], LM [26]) are based on an implicit assumption that if the user includes a term in a query, they expect the term to be found in a relevant document. Term weighting, therefore, typically focuses on estimating how useful a term will be (e.g., by modeling aboutness using term frequency and by modeling specificity using Inverse Document Frequency (IDF)), but without also modeling how central the term is to what the user is asking about [30]. In essence, current ranking functions typically defer to the user on the question of centrality. Hence, one of the important traits of almost all widely used ranking functions is that, all other things being equal, they will tend to prefer documents that contain all of the query terms [29] over documents that contain a dense collection of just some of those terms. For short queries, this is the right thing to do. For verbose queries, however, it is easy to believe that we could do better, and indeed some improvements have already been shown [19].

Research on the development of techniques tuned for verbose queries depends on the availability of test collections that actually contain verbose queries. While we might imagine many ways of obtaining such collections, the usual ap-

proach to date among information retrieval researchers has been to simply treat the description field of TREC topic descriptions as if they were actually issued by users as verbose queries. In general, the description field is intended to model what a searcher might first say to someone who will actually help them with their search, so this is not an unreasonable model. Because we are interested specifically in informational queries, which are fairly well modeled by TREC topics, we adopt the same approach to evaluation in this paper.

As a motivating example, consider the following queries involving a common term **effect**.

1. *How does Doppler ultrasound take advantage of the Doppler **effect** to create a moving image of the inside of the body?*

2. ***Effect** of temperature on measurement of alkaline phosphatase activity.*

When the collection being searched is the same, the keyword **effect** would be weighted equally if IDF weights were used. However, it seems clear to a human reader from the query context that the keyword **effect** is core to the information need expressed by the first query, while for the second query, it appears to be a high level concept. Hence, we would like to learn to give **effect** more weight in the first query.

To date, approaches to the problem of weighting (or, in the extreme case, selecting) terms to emphasize in long queries have essentially relied on a supervised learning framework. The key ideas behind such an approach are to compute a set of features that might be associated with good (or bad) results, and then to train a term weighting function based on those features to maximize some ranked retrieval evaluation measure. With enough training queries (for which relevance judgments are known), and with the right features, modest improvements can be obtained using this approach. The features are extracted from variety of sources such as the collection being searched, query log, or some other external resource such as Wikipedia [3].

In this paper, we adopt a different approach, drawing instead on an idea from text summarization, in which centrality is also key issue. In extractive text summarization, a key step is to select sentences that are either central to the a set of documents (for multi-document summarization), to a single document (for generic document summarization), or to a query (for query-based summarization). Among state of the art methods for this task, graph-based methods that iteratively estimate centrality using linear algebraic methods that converge to a fixed point vector that represents the centrality of every sentence have been widely adopted [11]. When graph-based techniques are applied for query-based summarization, the sentences are also somehow weighted (or selected) based on their similarity to the query [25]. The application of ideas from graph-based summarization to long queries is fairly direct, simply substituting term centrality for sentence centrality. Notably, although we need to train one free parameter to tune the relative contributions of traditional IDF and the query-specific term weights, the resulting technique is otherwise unsupervised.

One question that any such approach must answer is how to model query-specific term centrality. One way to integrate query context is to use the part of the collection that contains the original query terms densely and this can be achieved by simply using the top scoring documents in response to the original query measured by any reasonable ranking function. This is similar in spirit to the technique of Zhao and Callan [34], who extracted a set of features from the top returned documents and from the collection as a whole and then used supervised learning to predict query-dependent term necessity. We take the same starting point, but with an unsupervised approach. Operationally, the proposed algorithm is a two phase approach. In the first phase, the entire document collection is ranked using a standard bag of words retrieval model, and in the next step the top returned documents are used to compute the weight of query terms. Two key intuitions guided the development of the algorithm: (i) important terms are more frequent than the less important terms in the segment of the collection where original query terms are densely present and (ii) importance of a term increases if it is more frequent than other important terms. These two intuitions are then encoded in a PageRank-like link analysis algorithm to iteratively update the centrality weight using power iteration. Finally, the resultant term weight is computed by combining a centrality score with an IDF factor.

We evaluate the effectiveness of our method on a number of test collections with news and Web documents that together have relevance judgments for a large number of topics, each of which is described by a TREC *description* field. Three major conclusions can be drawn from our experiments. First, the proposed method consistently and significantly outperforms three progressively more capable baselines: (1) a simple query likelihood model, (2) a state-of-the-art sequential dependence model, and (3) a relevance model that learns to expand the query vocabulary based on the same initial search results but without any special handling for verbose terms. Second, we show that the effectiveness of our new unsupervised technique is competitive with state-of-the-art supervised algorithms that learn to rank documents for verbose queries from a large set of training queries on every test collection that we tried, even yielding statistically significantly improvements over the results of those algorithms in some cases. Third, we show that the our new method is computationally efficient, thereby making it practically employable in operational systems.

The remainder of the paper is organized as follows. Section 2 reviews prior research. Section 3 describes the proposed work. Section 4 and 5 present the experiment setup and results respectively. Finally, we conclude in Section 7.

2. PRIOR RESEARCH

Traditionally, discrimination between more and less important terms in a query has been done using statistical methods such as term frequency and IDF weighting, or (with language models) by using term-specific collection-frequency smoothing [13, 21]. We need to distinguish here between collection frequency or IDF on the one hand, either of which estimates the specificity of a *query* term in a collection, and term frequency on the other hand, which estimates the contribution of a term found in a *document* to describing the "aboutness" of that document. Our focus in this paper is on query terms, so we omit further review of the literature on term frequency (noting only that we use term frequency in our ranking models in the usual way).

Using supervised machine learning techniques for an automatic extraction of key concepts from documents was first

proposed by Turney [31], and later explored by several other researchers [14, 12]. Similar machine learning techniques have also proved beneficial for other tasks such as named entity recognition [6], content-targeted advertising [32] and summarization [16].

Key concept detection in verbose queries has been a subject of some previous work in information retrieval. Allan et al. [1] use a set of linguistic and statistical methods and a proximity operator to discover core terms (terms that must be present for a document to be relevant) in TREC *description* queries. Callan et al. [9] take a more knowledge-intensive approach, using noun phrases, named-entity recognition, exclusionary constraints, and proximity operators to convert TREC *description* queries into structured queries for the Inquery information retrieval system. Bendersky et. al [3] focus on weighting, estimating concept weights through a supervised learning technique, where each concept is represented using a set of features (such as concept term frequency in the collection, concept inverse document frequency, residual IDF [20], weighted information gain [35], Google n-gram frequency and the information from a query log containing 15 million query given to a commercial search engine. In that work, Ada-boost is used to estimate the weight of a concept from the features, and the resulting concept weights are then incorporated in a probabilistic language modeling framework. Similarly to Callan et al, their algorithm produces structured weighted queries, in this case using the Indri query language. All of these methods are supervised, in that feature weights must be learned.

Although these techniques can be employed with simple bag-of-words models, they have also been employed with more sophisticated techniques. Modeling query concepts through term dependencies has been shown to have a significant positive effect on retrieval effectiveness, especially for tasks such as Web search where relevance at high ranks is particularly critical. Recent research has shown that modeling query term dependencies and using non-uniform query term weights in a way more nuanced than IDF alone can significantly improve retrieval effectiveness, especially on very large collections and for long, complex queries. Metzler and Croft [23] develop a Markov Random Field (MRF) model to integrate the term proximity information in the language modeling framework to create a sequential dependence model. However, this method does not assign explicit weights to the query concepts. Bendersky et. al [4] extended the sequential dependence model for information retrieval by automatically learning query concept weights that simultaneously model query term dependencies and weighting generic query term concepts (e.g., unigrams, bigrams, etc.) in a unified, trainable framework. Once again, term proximity information, Google n-gram term statistics, query log and a number of retrieval corpus statistics are used in a supervised framework.

Lease [19] extended his previous work on term weighting to show that incorporating learned term weights in a sequential dependence model improves the retrieval effectiveness over the unweighted variant for verbose TREC *description* queries. The main argument was that the original MRF method estimates a parameter for each of its three feature classes from data, where parameters within each class are set via a uniform weighting scheme adopted from the standard unigram. Lease then showed that greater MRF retrieval accuracy can be achieved by better estimating within-class parameters, particularly for verbose queries employing natural language terms. A few additional lexical part-of-speech features are used in combination with the Markov Random Field and regression rank based learning algorithms.

All of the methods described so far focus on weighting only the terms that are actually present in the original query. Zhao and Callan [34], by contrast, performed a singular value decomposition on the top returned documents to identity synonyms of the query terms, then used those synonyms to predict term necessity. Like many of the previous methods, their necessity prediction method used a number of term-dependent and query-dependent features. Prior relevance judgments were used to compute the actual necessity values for the query terms as: $P(t|R) = r/R$, where r and $|R|$ denote the number of relevant documents that contain t and the number of relevant documents for the query. Each term is represented as a set of features $f_1, f_2 \dots f_n$ depending on the collection, the term t and the query q. A regression model M predicts necessity as a function of the features:

$$P(t|R_q) = M(q_1, q_2 \dots q_n) \quad (1)$$

Recently, Bendersky et. al [5] have introduced a parameterized query expansion method. The underlying motivation is that the methods that focus only on the query terms fail to take into account the latent concepts associated with the query. Hence, the queries are expanded by identifying additional terms, and they are weighted using a supervised parameterized model that employs a number of features similar to the many of the methods described before. Query expansion raises questions of computational cost, however, since using expansion terms generates additional disk activity. In our work we therefore focus on methods that do not introduce expansion terms.

Graph based methods are widely applied in term weighting, smoothing language models and search result reranking tasks. Collins-Thompson [10] propose and evaluate a Markov chain based framework for modeling combination of term relations (such as synonym, morphological variants, co-occurring term) and apply the model to query expansion. Given a small set of initial query terms, the method constructs a term network and use a random walk to estimate the likelihood of relevance for potential expansion terms. The random walk model uses features that can come from a variety of sources, such as term co-occurrence in an external collection, Co-occurrence in the top retrieved documents, synonym dictionaries, general word association scores. Mei et al. [22] propose a generalized optimization framework for smoothing language model using document-document and word-word graphs. Graph created from top returned documents has been applied to improve the precision of search results [17]. This method ranks the documents based on their pagerank score, where the links between every pair of document are created using KL-divergence of their unigram language model. However, none of these methods focus on query term re-weighting.

3. PROPOSED METHOD

Each of the methods discussed in the previous section used some form of supervised learning, with some also using external resources (e.g., Wikipedia). In contrast, we develop an unsupervised method that uses only the collection on which retrieval is to be performed. In this section, we first describe the problem and the retrieval framework. We then

describe our proposed centrality based algorithm to compute the weight of the search terms. Finally, we analyze computational efficiency to show that the proposed graph based algorithm is not computationally expensive.

3.1 Overview

Given a query $Q = q_1 q_2 \dots q_n$, the main goal of our proposed algorithm is to determine the importance of each of the query terms. Inverse document frequency is widely used to achieve this goal. However, IDF is not capable of factoring in the query context and hence it treats a term as equally important irrespective of the query in which it appears. Thus, the major goal of our algorithm is to compute weights for query terms that adapt to query. As a consequence, with our technique, even if a term t_1 is rarer than t_2 in the collection, t_2 might still enjoy preference over t_1 if other query terms provide evidence for its importance in a specific query. The query information is implicitly integrated by taking term occurrence in top-ranked documents as a source of evidence. The underlying rationale is that the documents which are returned in response to the original query tend to focus on the intent of the query and hence the distribution of the terms in that part of the collection can provide somewhat more reliable evidence for identifying useful terms than would the collection as a whole.

The key assumption that motivates the design of the proposed algorithm is that a query term which is more frequent than the other query terms in the part of collection where query terms are dense is likely to be more important than the others. At the same time, a central term is more likely to be frequent than other central terms. Hence, the centrality of terms can be considered as the major evidence of term importance, while the relative frequency is the basis for the centrality measure.

Formally, our goal is the following. Given a query $Q = q_1 q_2 \dots q_n$ and document D, the similarity between query and document is defined as

$$S(Q, D) = \sum_{i=1}^{n} I(q_i) \cdot W(q_i, D) \tag{2}$$

where $W(t, D)$ can be any reasonable weighting function such as log-likelihood Dirichlet prior language model [33] or BM25. In this paper we use Dirichlet language model to compute $W(q_i, D)$ as given below

$$W(q_i, D) = \log\left(\frac{c(q_i, D) + \mu \frac{c(q_i, C)}{|C|}}{|D| + \mu}\right) \tag{3}$$

where $c(q_i, D)$ and $c(q_i, C)$ are the count of q_i in document D and collection C respectively; μ is the free parameter; $|C|$ and $|D|$ are the number of terms in the collection and document respectively. Our main goal is to compute the value of $I(q_i)$. The section that follows describes the proposed algorithm.

3.2 Algorithm

We now turn to describe our main algorithm for estimating term importance. We reiterate that our goal is to compute the global importance (in the sense that the weights are query dependent and document independent) of the query terms with respect to the original query. The query dependency is integrated via the ranked list generated by running the original query using a standard bag of words model (in our case language model with Dirichlet prior).

The proposed algorithm stands mainly on the following two key hypotheses

1. *A term t is important if it is more frequent than the other terms in the query-relevant part of the collection. The query-relevant part is defined as the top returned documents upto a pre-specified cut-off in response to the initial query.*

2. *A term t is more important if it is more frequent than other important terms in the query-relevant part of the collection.*

The first hypothesis is the classic term frequency hypothesis in information retrieval that is the basis of all known term importance scheme. On the other hand the second hypothesis is intended to reward the terms which are associated with other important terms. The second hypothesis, in particular, dictates the need for a recursive definition. Hence, the above two hypotheses can be mathematically formulated as

$$A(t_i) = CumRF(t_i|t_j) \cdot A(t_j),\ t_i \neq t_j \tag{4}$$

where $A(t)$ denotes the centrality of the term t and $CumRF(a|b)$ denotes the cumulative frequency of a relative to the frequency of b in the set of selected documents, D. Hence, $CumRF(a|b)$ is defined as

$$CumRF(a|b) = \sum_{d \in D} RF(a|b, d) \tag{5}$$

where $RF(a|b, d)$ is computed as follows

$$RF(a|b, d) = \begin{cases} \frac{\log_2(1+c(a,d))}{\log_2(1+c(b,d))}, & \text{if } c(b, d) > 0 \\ \log_2(1 + c(a, d)), & \text{otherwise} \end{cases} \tag{6}$$

where $c(a, d)$ is the frequency of the term a in document d. In Equation 6, 1 is added primarily to address zero frequency problem. Also note that we did not use raw term frequency in Equation 6, instead we apply a logarithmic damping function to restrict the contribution of very high frequency terms, since raw term frequency is known to be problematic [28]. Indeed, we also experimented with raw term frequency and found that logarithmic damping was always better. For space constraint we omit those results.

We now extend Equation 4 to measure the importance of the term t by taking into account all other query terms as follows

$$A(t_i) = \sum_{j=1, i \neq j}^{|q|} CumRF(t_i|t_j) A(t_j) \tag{7}$$

Therefore, for each query term t_i, we have Equation 7. Thus, if the query contains n terms, we have n such equations. Hence, this system of equations can be written compactly using a matrix notation as

$$\mathbf{A}^T = \mathbf{CumRF} \cdot \mathbf{A}^T \tag{8}$$

where $\mathbf{I}$ is the term importance vector and $\mathbf{CumRF}$ is the matrix containing pairwise relative frequencies whose (i, j)-th entry is given by $CumRF(t_i|t_j)$.

From Equation 8, it is clear that the vector $\mathbf{A}^T$ is the principal eigenvector of the matrix $\mathbf{CumRF}$. We use power

iteration to compute the vector $\mathbf{A}^T$. Power iteration [8] is a well known technique to find only the principal eigenvector and has been used extensively in many link analysis algorithms [7] (PageRank, for example) for Web search. The power iteration starts by setting all the entries of the vector $\mathbf{A}^T$ to 1 and then iterates a specified number of times. In fact, our experiments suggest that 7-12 iterations are sufficient to reach convergence.

Integrating IDF.

So far we have used relative frequency of terms in the top returned documents in order to determine the centrality of a term reflecting its importance. However, this evidence alone may not be always helpful, since terms having moderate generality in the collection tend to occur in many documents. Hence, they have relatively higher chance of occurrence even in a randomly chosen document. As a result, reliance only on the frequency of occurrence may sometimes give undue credit to those terms which are present in many documents even with lesser frequency than the terms that happen to be important. In order to prevent those terms from being considered as a potentially useful candidate, we use inverse document frequency information along with the centrality score.

However, using the full contribution of the standard IDF function would not be an appropriate choice because collection-specific term weights already contain a term specificity component.[1] What is required therefore, is a function that transforms IDF values such that the rare terms are rewarded, but with diminishing effect. We can achieve this goal approximately the right way using the following simple function:

$$didf(t) = \frac{idf(t)}{c + idf(t)} \tag{9}$$

where $idf(t)$ is the standard $log(\frac{N}{df})$ (N is the number documents in the collection) and c is the free parameter. The function in Equation 9 is an increasing function of idf and the parameter c can be set to regulate the contribution made by IDF. A value of c closer to zero essentially removes IDF from the computation (i.e $didf(t) \rightarrow 1$), while a very large value of c would make $didf(t)$ essentially undamped. Since neither extreme is desirable, the value of c should not be too small or too large.

The resultant weight of each of the query term q_i is computed by multiplicatively combining the centrality score $A(q_i)$ and $didf(q_i)$. Formally:

$$I(q_i) = A(q_i) \cdot didf(q_i) \tag{10}$$

3.3 Efficiency Issues

Note that the algorithm described in the preceding section must be run at query time, and therefore it is important that it be computationally efficient. In this section we explore that aspect of the proposed algorithm and quantify the computation time required for query processing. We look at this problem from two different perspectives: the time required for query term re-weighting, and the additional computation required for re-ranking the collection.

Note that we have used power iteration to compute the term centrality scores. Two approaches are prevalent in practice as a terminating condition: convergence, or a fixed number of iterations. We chose a fixed number of iterations (10), since after 10 iterations the difference between two successive eigenvector estimates is negligible.

In each iteration we need to do n^2 computations, where n is the order of the matrix, which is the length of the query. For k iterations, the computational complexity is

$$k \times n^2 = O(n^2).$$

A typical verbose query in our experiments is around 15-20 words. At that scale, the resulting computation is quite fast. For applications with hundreds of query terms, we might need to give some attention to heuristic preselection of terms, however.

Another important efficiency issue that needs to be addressed is the cost associated with the re-ranking of whole collection using re-estimated term weights. Note that, unlike query expansion methods, our proposed method does not add any new terms, thus the re-ranking step does not need to compute the score of any documents other than those which had been found to include at least one query term in the first phase of ranking. Hence, the inverted lists for all query words which were fetched during the first phase of ranking can be stored into memory for the next phase of processing, thereby obviating the need for reading posting files from the disk again. Moreover, it is possible that some terms may receive zero or negligible query-specific term weights, and such terms can be safely ignored when re-ranking is done. The net effect of these factors is that the in-memory re-ranking computations are far faster than the initial search.

Table 1: Test collections used in our experiments

Collection Name	# docs	Topics
TREC 6,7,8	528,155	301-450
WT10G	1,692,096	451-550
GOV2	25,205,179	701-850
ClueWeb-B	50,220,423	1-100

4. EXPERIMENT SETUP

In this section, we describe the experimental setup for evaluating our proposed algorithm. We provide a summary of the test collections used for our experiments in Table 1. We note that collections vary both by type (TREC 6,7,8 is a newswire collection, while WT10G, GOV2 and ClueWeb-B are Web collections), and by number of documents and number of topics, thus providing a diverse experimental setup for assessing robustness. GOV2 is a homogeneous web collection from .gov domain, while ClueWeb-B is a heterogeneous web collection used in recent TREC web tracks.

All indexes and topics were stopped using a standard stopword list and stemmed using a Porter stemmer. Since our goal is to address verbose queries, the *description* portions of the TREC topics were used to construct the queries, following previous work on verbose queries [3, 19]. A Dirichlet language model was used as the primary ranking function. Statistically significant differences are determined using two sided paired t-test at a 95% confidence level.

To measure the retrieval quality we use two metrics: Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain to depth 20 (NDCG@20) [15]. Notably, NDCG leverages graded relevance judgments, and thus is suitable for the Web collections (WT10G, GOV2, ClueWeb).

[1]The same would be true for the use of collection frequency statistics for smoothing with language models

4.1 Baselines

We compare the effectiveness of our methods with a number of unsupervised and supervised baselines, including the standard bag of words query likelihood Dirichlet language model. The detailed descriptions of the baselines are as follows:

1. Query likelihood using Dirichlet language model (QL).
2. RM3 [18] relevance model for query term re-weighting (RW-RM3).
3. Sequential Dependence model (SD) [23].
4. Key Concept model (KC) [3].
5. Weighted Sequential Dependence model (WSD) [4].

RM3 is the relevance model pseudo-relevance feedback algorithm implemented in the language modeling framework. Since we are only interested in re-weighting the query terms, our implementation of RM3 is used to compute only query term weights. RW-RM3 interpolates the probability of the terms in original topic with the probability of the terms computed from the top returned documents.

The Sequential Dependence model is a state-of-the-art ranking function based on Markov Random Field. In addition to a standard bag-of-words score, ordered and unordered query term proximity are used as additional evidence for relevance. It has been shown that the term proximity information is able to capture some measure of term dependence, which can be useful for long complex queries.

The Key Concept and Weighted Sequential Dependence models are supervised methods. Both of these methods use a number of collection and query dependent features (e.g., residual IDF, frequency of query concepts in a query log, and Wikipedia articles) to learn the importance of search terms. Key concept uses Ada-boost, while WSD uses coordinate-level ascent algorithm for parameter estimation.

Additionally, to characterize the effect of our decision not to add additional query terms, we compare the performance of the proposed method with two state of the art query expansion methods (relevance model and latent concept expansion [24]) that adds 5 highly selective terms that are found in the top-ranked documents to the original query.

The proposed method, and all the baselines, contain one or more free parameters that influence the retrieval quality. For fair comparison, the free parameters of all the methods are set using five-fold cross-validation, optimizing MAP.

5. RESULTS

In this section we present effectiveness measures for our proposed algorithm and for the baselines. We begin in Section 5.1 by comparing to unsupervised baselines. Section 5.2 then compares to the two supervised methods. The remaining subsections provide analyses of specific system variants.

5.1 Comparison to Baseline Methods

Table 2 compares the effectiveness of our proposed method (TA) with that of the three unsupervised baselines. The baselines include Query Likelihood model that ranks the documents using a Dirichlet language model (QL), query term Re-Weighting using a Relevance Model (RW-RM3) and the Sequential Dependence model (SD). QL and RW-RM3 are completely unsupervised, while SD can be considered moderately supervised method because of a number of parameters need to be estimated from training data.

Table 2 clearly shows that TA always outperforms all three baselines when effectiveness is measured using MAP. On the TREC news collection, TA yields a 14% improvement over the best unsupervised technique (RW-RM3), and all of the improvements are statistically significant. Similar conclusions can be drawn from all three Web collections. Once again, TA is statistically significantly better than each baseline on every collection by between 6% and 26%, with the largest improvements on the most modern test collection (ClueWeb-B). Query term reweighting using RM3 (RW-RM3) is not an effective approach, since only on TREC-678 RW-RM3 gives statistically significant MAP over QL. SD provides significant benefit over QL on TREC-678 and GOV2.

Similar results are seen with NDCG@20, with statistically significant improvements between 6% and 26% over each baseline for all three of the Web collections. The improvement over QL for the TREC collection is smaller (8%), but still statistically significant. No statistically significant difference is observed for NDCG@20 between TA and either RW-RM3 or SD. This one difference from the results with MAP is attributable either to the NDCG cutoff at 20 or to differences in the discount rate between the two measures. Performance of RW-RM3 is roughly comparable to QL, while SD is significantly better than QL on three out of four collections.

In summary, our results indicate that our proposed centrality-based algorithm yields statistically significant improvements in retrieval accuracy over both a reweighting-only relevance model and a sequential dependence model by both measures for Web collections, and by MAP for the TREC collection.

5.2 Comparison to Supervised Methods

In the previous section we compare the effectiveness of TA with a number of unsupervised baselines. In this section we compare to two supervised baselines. We reproduce the retrieval results for KC using queries made available by Bendersky et al [3] for three of our collections; we have no KC queries for ClueWeb-B so we do not report KC results on that collection. We have reimplemented the WSD model ourselves because queries for that model are not available to us, but we note that because we do not have access to the query log used by Bendersky et al [4] we omit the features that depend on a query log. Improvements that we report over WSD may therefore somewhat overstate what would be seen were WSD to be implemented with access to a rich and representative query log.

Table 3 summarizes the effectiveness of KC, WSD and TA on the same four collections. TA statistically significantly outperforms KC on all three collections for which we can compute KC results, by between 7% and 15%. TA also statistically significantly outperforms WSD on three collections, by between 8% and 13%, although no statistically significant difference was detected between TA and WSD on GOV2. The results for NDCG@20 give a slightly different picture, with only ClueWeb-B showing a statistically significant difference (with TA significantly better than WSD; KC can not be run on ClueWeb-B). Again, differences between MAP and NDCG@20 could results from the cutoff at 20 or from differences in the discount rate. The results for MAP suggest that KC and WSD are always significantly bet-

Table 2: Retrieval effectiveness of the proposed method (TA) compared to unsupervised baselines. Statistically significant improvements are indicated using the first letter of the less effective method. The highest value per column is bolded. The numbers in parenthesis indicate relative improvement over QL, RW-RM3 and SD, respectively. NDCG is NDCG@20.

Metric	Method	TREC	WT10G	GOV2	ClueWeb
MAP	QL	0.191	0.184	0.256	0.124
	RW-RM3	0.202^{q}	0.194	0.263	0.128
	SD	0.200^{q}	0.193	0.276^{q}	0.129
	TA	$\mathbf{0.230}^{qrs}$(20%, 14%, 15%)	$\mathbf{0.231}^{qrs}$(25%, 19%, 20%)	$\mathbf{0.292}^{qr}$(14%, 11%, 6%)	$\mathbf{0.156}^{qrs}$(26%, 22%, 21%)
NDCG	QL	0.381	0.318	0.420	0.179
	RW-RM3	0.390	0.323	0.426	0.186
	SD	0.402^{q}	0.340^{q}	0.432^{q}	0.188
	TA	$\mathbf{0.413}^{q}$(8%, 6%, 3%)	$\mathbf{0.365}^{qrs}$(15%, 13%, 7%)	$\mathbf{0.451}^{qrs}$(7%, 6%, 4.4%)	$\mathbf{0.225}^{qrs}$(26%, 21%, 20%)

Table 3: Retrieval effectiveness of our proposed method (TA) compared to supervised baselines. Statistically significant differences are indicated using the first letter of the less effective method. The highest value per column is bolded. Numbers in parenthesis indicate relative improvement over KC and WSD respectively.

Metric	Method	TREC	WT10G	GOV2	ClueWeb
MAP	QL	0.191	0.184	0.256	0.124
	KC	0.212^{q}	0.201^{q}	0.273^{q}	-
	WSD	0.207^{q}	0.214^{q}	0.285^{q}	0.138^{q}
	TA	$\mathbf{0.230}^{kw}$ (9%, 11%)	$\mathbf{0.231}^{kw}$ (15%, 8%)	$\mathbf{0.292}^{k}$ (7%, 3%)	$\mathbf{0.156}^{w}$ (-, 13%)
NDCG@20	QL	0.381	0.318	0.420	0.179
	KC	0.382	0.330^{q}	0.432	-
	WSD	$\mathbf{0.416}^{q}$	0.360^{q}	0.439^{q}	0.188
	TA	0.413 (8%, -1%)	$\mathbf{0.365}^{k}$ (11%, 1%)	**0.451** (4%, 4%)	$\mathbf{0.225}^{w}$ (-, 20%)

ter than QL, while no significant difference has been found between KC and WSD. The results measured in terms of NDCG@20 reveal that WSD is significantly better than QL on TREC-678, WT10G and GOV2, while KC is significantly better than QL only on WT10G collection.

5.3 Effect of Term Centrality

Recall that the proposed algorithm stands on two assumptions. The first assumption is based on the importance of document specific frequency of one term relative to another, while the second assumption takes into account the centrality of the terms based on their relative frequency in the top ranked documents. So far, we have examined the effect of these two assumptions together. The experiments in this section are designed to understand the effect of term centrality (i.e, the iterative update of term weights) in determining the importance of a query term.

To that end, we carry out three set of experiments for each of the test collections. The first set of experiments is done using only the query likelihood model (no query specific weighting). The second set of experiments incorporates the weight of the query terms based on the relative frequency and the IDF factor only (without iterative weight update for term centrality), while the third set of experiments uses the weights that are computed using the full centrality based algorithm and the IDF factor (equation 10). For all the experiments, the number of top ranked documents is set to 20, as in our main experiments.

Table 4 shows the experiment results, which clearly show that our full term centrality iteration method (TA) results in improved effectiveness over No Centrality iteration (NC) on all four test collections of between 7% and 16% when measured by MAP. When measured by NDCG@20, statistically significant improvements result from the power iteration step for all three Web collections, but no statistically significant difference between TA and NC was observed with NDCG@20 for the TREC collection. Because we believe MAP is well suited as an evaluation measure for the TREC collection (which has binary relevance judgments), we do not find this one difference troubling.

5.4 Effect of Number of Documents

We reiterate that the top ranked document in response to the original query is the only source of information for the proposed weighting algorithm. Hence, an important issue with the proposed method is the proper choice of the number of top ranked documents as a source of weight estimation. In this section we specifically investigate the following questions on document selection.

1. What is a good value of number of documents?
2. Is the number of documents (that leads to optimal effectiveness) consistent across collections?
3. Does the effectiveness fluctuate with the number of documents and to what extent?

To answer the above three questions, we conduct experiments on all the four test collections we have used to evaluate our method. Our experimental methodology estimates the query term weight by varying the number of documents from 5 to 50 in the increment of 5. The collections are then ranked and evaluated using two standard metrics, NDCG@20 and MAP.

Figure 1 and 2 show the impact of number of documents on retrieval quality. As it turned out, the best effectiveness measured in terms of NDCG@20, often comes when the number of documents is between 15 and 25. On TREC and WT10G, TA gains the maximum NDCG@20 when the

Table 4: Impact of term centrality on retrieval effectiveness. NC and TA denote the effectiveness weighting without centrality and full fixed point centrality weighting (equation 10), respectively. The highest value per row is bolded. Numbers in parenthesis indicate relative improvement over QL and NC respectively. Superscripts q and n denote statistically significant improvement over QL and NC respectively.

		TREC	WT10G	GOV2	ClueWeb
MAP	QL	0.191	0.184	0.256	0.124
	NC	0.215^{q} (13%)	0.217^{q} (17%)	0.270^{q} (6%)	0.134 (8%)
	TA	$\mathbf{0.230}^{qn}$ (20%, 7%)	$\mathbf{0.231}^{qn}$ (26%, 7%)	$\mathbf{0.292}^{qn}$ (14%, 8%)	$\mathbf{0.156}^{qn}$ (26%, 16%)
NDCG@20	QL	0.381	0.318	0.420	0.179
	NC	0.396 (4%)	0.334^{q} (5%)	0.431 (3%)	0.197^{q} (10%)
	TA	$\mathbf{0.413}^{q}$ (8%, 4%)	$\mathbf{0.365}^{qn}$ (15%, 9%)	$\mathbf{0.451}^{qn}$ (7%, 5%)	$\mathbf{0.225}^{qn}$ (26%, 14%)

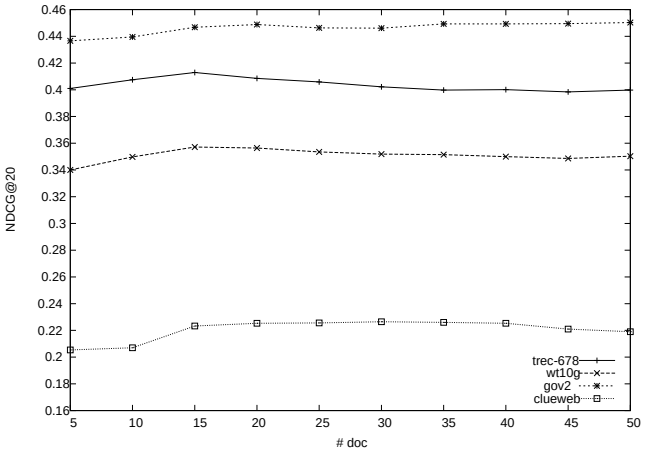

Figure 1: Effect of number of documents: NDCG.

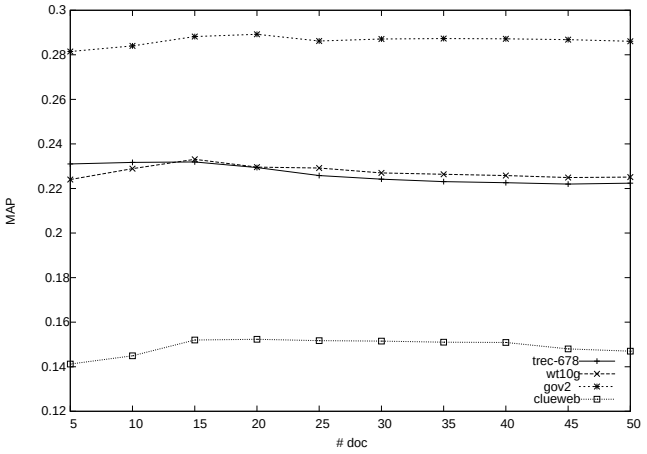

Figure 2: Effect of number of documents: MAP.

Table 5: Computation time (in minutes).

	TREC	WT10G	GOV2	ClueWeB
RW-RM3	**0.6**	**1.0**	**8.2**	**14.1**
SD	0.9	1.5	16.3	21.0
KC	0.7	1.2	9.8	15.7
WSD	1.2	1.5	22.7	28.3
TA	0.6	1.1	8.6	14.6

number of documents is set to 15, while for both the GOV2 and ClueWeb collections the value is 20. We can draw similar conclusions based on the MAP values (right figure). Once again, the numbers of documents for optimal MAP for four collections are very consistent with those for optimal NDCG@20.

Figures 1 and 2 depict other two important facts. First, the effectiveness fluctuates only marginally with the number of documents, and the trend is quite similar for different collections. Second, to the extent a peak can be discerned, somewhere in the range of 15–25 documents seems like a good choice, both NDCG@20 and for MAP.

5.5 Computation Time

In Section 3.3 we mathematically analyzed the computation time of the proposed method. We have argued that even if the proposed algorithm (TA) must be run at query time and uses a graph based method to compute the term importance, TA is still computationally reasonably efficient. In this section we report the actual time taken by the proposed method and compare the same with the baseline methods.

The experiments were conducted on a stand-alone desktop computer with 16GB of main memory. On each collection, each method was run 5 times; the reported computation time is the average over those 5 runs. For TA and RW-RM3, we set the number of top returned documents to 20 based as in our previous experiments.

Table 5 presents the computation time taken by all the methods on four test collections used in our experiments. We reiterate that TA and RW-RM3 build the queries online, while KC and WSD build the queries offline. That is the principal reason we are interested in comparing the efficiency of TA to KC and WSD. Table 5 reflects various facts. First, RW-RM3 and TA, despite being two pass online methods, outperform the other three (KC, SD, WSD) single pass methods. Second, although, RW-RM3 is more efficient than TA, the difference is negligible. Note that RW-RM3 and TA differ only in the weight estimation process, and thus this marginal difference corroborates the mathematical analysis

presented in Section 3.3. Third and more interestingly, KC, SD and WSD have been found to be computationally more complex than TA, despite being single pass methods. The reason for the inefficiency is that SD and WSD use ordered bi-grams and the unordered positional proximity of consecutive query terms in the documents up to some window size (8, in our implementation). Thus, the efficiency of both SD and WSD depends not only on the size of the postings lists for each query term, but also on the frequencies of the query terms in each document. KC, by contrast, does not use positional information, but it produces structured Lemur queries consisting of multiple subqueries, and a single query term can often appear in more than one subquery, even when the term is present only once in the original query. We believe these are the main reasons why KC, SD and WSD are slower than TA and RW-RM3.

In summary, the proposed unsupervised method (TA) not only often gives significant retrieval effectiveness over the state of the art supervised baselines (and unsupervised as well), TA is computationally more efficient than supervised methods. Although, RW-RM3 is comparable to TA in terms of efficiency, the retrieval performance of RW-RM3 is almost always significantly poorer than TA.

5.6 Comparison to Query Expansion

All of our comparisons to this point have been with techniques that reweight query terms, but that do not add new terms to the query. In this section we explore the effect of that restriction by implementing the RM3 relevance model and the Latent Concept expansion Model (LCE), both of which add 5 new terms to each query. LCE uses a sequential dependence model, and thus is more expensive than RM3. We set number of feedback documents to 20. The interpolation parameter was set using five-fold cross-validation. As Table 6 shows, th EX-RM3 (RM3 with Expansion) model does not markedly change the pattern of results when measured by either MAP or NDCG@20. Specifically, we see no statistically significant difference between EX-RM3 and RW-RM3 for any collection by NDCG@20, and we see that all but one of the statistically significant improvements of TA over RW-RM3 are present with EX-RM3 as well. The one exception is WT10G, the smallest of the three Web collections, where when measured by MAP, TA is statistically significantly better than RW-RM3 but is statistically indistinguishable from EX-RM3. Because we believe NDCG@20 is the better measure for Web collections, we do not find this one difference troubling. Differences in mean values might lead us to believe that LCE could be a better choice than EX-RM3, but the difference is statistically significant in only one of the eight cases (two measures, four collections). We also note that EX-RM3 is 1.4 times slower than TA while RW-RM3 is generally slightly faster than TA, confirming our belief that reweighting offers efficiency gains over expansion.

5.7 Parameter Sensitivity

Our proposed weighting scheme contains one free parameter, c, which regulates the contribution of the IDF factor to the final weight. In these experiments we seek to gain insight into the effect of that parameter on retrieval effectiveness. We conduct a set of experiments on each collection by varying the value of c from 1 to 20 in increments of 1. Once again, we set the number of documents to 20, as in the previous experiments.

Table 6: Effect of query expansion. Statistically significant differences are indicated using the first letter of the less effective method. The highest value per column is bolded. NDCG is NDCG@20.

		TREC	WT10G	GOV2	ClueWeb
MAP	QL	0.191	0.184	0.256	0.124
	RW-RM3	0.202^{q}	0.194	0.263	0.128
	EX-RM3	0.224^{qr}	0.212^{qr}	0.276^{q}	0.122
	LCE	$\mathbf{0.232}^{qr}$	0.222^{qr}	0.282^{qr}	0.131^{q}
	TA	0.230^{qr}	$\mathbf{0.231}^{qre}$	$\mathbf{0.292}^{qre}$	$\mathbf{0.156}^{qrel}$
NDCG	QL	0.381	0.318	0.420	0.179
	RW-RM3	0.390	0.323	0.426	0.186
	EX-RM3	0.403^{q}	0.338^{q}	0.428	0.175
	LCE	$\mathbf{0.428}^{qre}$	0.340^{q}	0.444^{qr}	0.182^{q}
	TA	0.413^{q}	$\mathbf{0.365}^{qrel}$	$\mathbf{0.451}^{qre}$	$\mathbf{0.225}^{qrel}$

Figure 3 shows the impact of different values of c on MAP. For the ClueWeb-B collection, TA achieves its highest MAP at $c = 9$. The GOV2 collection also exhibits a noticeable peak, in this case $c = 11$ is the best choice. Little effect on MAP is evident for mid-range values of c on TREC or the WT10G collections. Overall, we conclude that values of c between 8 and 12 seems to be suitable choices.

6. ERROR ANALYSIS

Now looking in more detail at queries that are significantly hurt by TA when compared to the QL baseline, we can identify two causes. First, queries in which few relevant documents actually exist necessarily result in low precision initial searches for the fixed cutoff at 20 that we used, which in turn results in too little useful evidence to guide our method. Second, high IDF terms that occur highly frequently in the top returned documents can exercise undue influence. One way to address this would be to adjust the term frequency damping function (presently $\log(1 + tf)$) to something with an even slower growth rate.

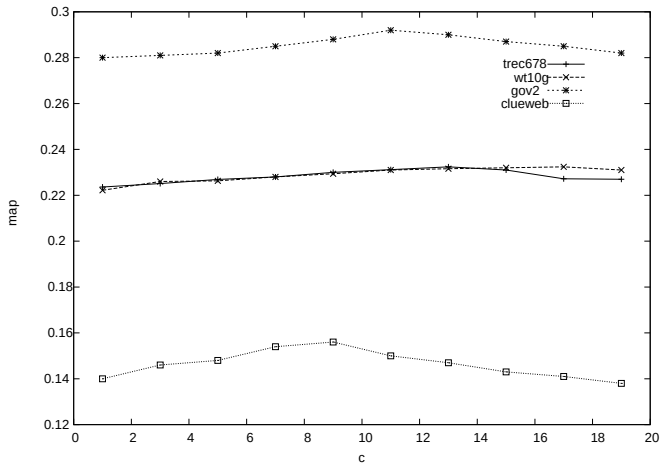

Figure 3: Effect of parameter c

7. CONCLUSION

We have introduced an unsupervised method for weighting query terms in ways that reflect both the specificity of the term, as in traditional measures, and the centrality of the term. We have shown that this new method to be as effective as the best presently known supervised method when tested on TREC collections using description queries, and that in several cases we obtain statistically significant improvements over the best known techniques. Moreover, we have shown that our method is computationally more efficient than the state of the art supervised approaches. As a next step, we plan to now develop test collections that more closely model specific verbose information query applications such as hands-free/eyes-free in-vehicle search or example-based queries in exploratory search settings.

Acknowledgments

This research was supported in part by DARPA contract HR0011-12-C-0015 and NSF award 1065250.

8. REFERENCES

[1] J. Allan, J. Callan, B. C. L. Ballesteros, B. Croft, L. Ballesteros, J. Broglio, J. Xu, and H. Shu. Inquery at TREC-5. In *TREC-5*, 1997.

[2] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM TOIS*, 20(4), 2002.

[3] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR*, 2008.

[4] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *WSDM*, 2010.

[5] M. Bendersky, D. Metzler, and W. B. Croft. Parameterized concept weighting in verbose queries. In *SIGIR*, 2011.

[6] D. M. Bikel, R. Schwartz, and R. M. Weischedel. An algorithm that learns what is in a name. *Machine Learning*, 34(1–3), Feb. 1999.

[7] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Link analysis ranking: Algorithms, theory, and experiments. *ACM TOIT*, 5(1), 2005.

[8] C. Brezinski and M. Redivo-Zaglia. The Pagerank Vector: Properties, computation, approximation, and acceleration. *SIAM J. Matrix Anal. Appl.*, 28(2), 2006.

[9] J. P. Callan, W. B. Croft, and J. Broglio. TREC and TIPSTER experiments with Inquery. In *IPM*, 1994.

[10] K. Collins-Thompson and J. Callan. Query expansion using random walk models. In *CIKM*, 2005.

[11] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479, 2004.

[12] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI*, 1999.

[13] D. Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval: The importance of a query term. In *SIGIR*, 2002.

[14] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*, 2003.

[15] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM TOIS*, 20(4), 2002.

[16] K. Knight and D. Marcu. Statistics-based summarization - step one: Sentence compression. In *AAAI*, 2000.

[17] O. Kurland and L. Lee. Pagerank without hyperlinks: Structural re-ranking using links induced by language models. In *SIGIR*, 2005.

[18] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR*, 2001.

[19] M. Lease. An improved Markov Random Field model for supporting verbose queries. In *32nd SIGIR*, 2009.

[20] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001.

[21] Q. Mei, H. Fang, and C. Zhai. A study of poisson query generation model for information retrieval. In *SIGIR*, 2007.

[22] Q. Mei, D. Zhang, and C. Zhai. A general optimization framework for smoothing language models on graph structures. In *SIGIR*, 2008.

[23] D. Metzler and W. B. Croft. A Markov Random Field model for term dependencies. In *SIGIR*, 2005.

[24] D. Metzler and W. B. Croft. Latent Concept Expansion Using Markov Random Fields. In *SIGIR*, 2007.

[25] J. Otterbacher, G. Erkan, and D. R. Radev. Biased Lexrank: Passage retrieval using random walks with question-based priors. *Information Processing and Manageement*, 45(1), 2009.

[26] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR*, 1998.

[27] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, 1994.

[28] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *IPM.*, 24(5), 1988.

[29] A. Singhal. Modern Information Retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[30] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *SIGIR*, 1996.

[31] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4), 2000.

[32] W.-t. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *WWW*, 2006.

[33] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM TOIS*, 22(2), 2004.

[34] L. Zhao and J. Callan. Term necessity prediction. In *CIKM*, 2010.

[35] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *SIGIR*, 2007.