Suggesting Topic-Based Query Terms as You Type

Ju Fan [1], Hao Wu [2], Guoliang Li [3], Lizhu Zhou [4]

Department of Computer Science and Technology, Tsinghua University
Tsinghua University, Beijing 100084, China
{ [1]fan-j07, [2]haowu06}@mails.tsinghua.edu.cn
{ [3]liguoliang, [4]dcszlz}@tsinghua.edu.cn

***Abstract*— Query term suggestion that interactively expands the queries is an indispensable technique to help users formulate high-quality queries and has attracted much attention in the community of web search. Existing methods usually suggest terms based on statistics in documents as well as query logs and external dictionaries, and they neglect the fact that the topic information is very crucial because it helps retrieve topically relevant documents. To give users gratification, we propose a novel term suggestion method: as the user types in queries letter by letter, we suggest the terms that are topically coherent with the query and could retrieve relevant documents instantly. For effectively suggesting highly relevant terms, we propose a generative model by incorporating the topical coherence of terms. The model learns the topics from the underlying documents based on Latent Dirichlet Allocation (LDA). For achieving the goal of instant query suggestion, we use a trie structure to index and access terms. We devise an efficient top-k algorithm to suggest terms as users type in queries. Experimental results show that our approach not only improves the effectiveness of term suggestion, but also achieves better efficiency and scalability.**

I. Introduction

In an age of information explosion, web search engine has become more and more important to help users access a huge volume of web pages. However, the imprecise and low-quality queries have been reported as one of the main reasons of unsatisfying performance of search engines [1]. Generally, users cannot exactly provide the specification of their information needs, and even the well-formulated queries may lead to few hits, because users have limited knowledge of the underlying documents and may type in mismatched keywords. The uncommon, nonspecific and sometimes ambiguous queries always lead to non-relevant and even puzzling results [2], [3].

Query term suggestion, also known as *Interactive Query Expansion* (IQE), is an indispensable technique to effectively address the above problem. It aims to help formulate high-quality queries by presenting a list of candidate terms for users to choose during the retrieval session. The selected term is then added to the initial query for the further specification of information needs. Some user studies show that this technique can improve the retrieval effectiveness significantly [4], [5]. Unfortunately, developing an effective IQE system is quite challenging, because it is difficult to predict which terms would be suggested. Early works generate the candidate terms by means of relevance feedback and manually constructed thesauruses [6], [7]. Recently, some works focus on discovering the relationship of terms in general dictionaries (e.g., WordNet) [8], knowledge bases (e.g., Wikipedia) [9], underlying documents [10], [11], and query logs [12]. However, existing studies neglect an important factor, the topic information, which is shown crucial for specifying the user's information needs [13]. Intuitively, a document can be seen as a mixture of some topics in the semantic level, and a user, who issues a particular query, is generally interested in a single or a small number of topics. Take the publication search on the DBLP data set[1] as an example. In each retrieval session, a user wants the publication records focusing on some specific topics (e.g., web search, databases, etc.) rather than a rough-and-tumble of documents with different topics. Thus, queries that reflect the user's topical intention perform well to retrieve topical relevant and high-quality documents [2].

Based on these observations, we propose a novel interactive search feature to suggest *topic-based* query terms as a user types: given a sequence of previous query terms as the *context* and a currently input *prefix*, a list of terms with the prefix that not only are topically coherent with the context but also can lead to good hits are suggested instantly. Moreover, the best hit documents of the terms are also grouped and displayed for providing users more information about the underlying data. Fig. 1 gives a screenshot of a system using our proposed techniques on the DBLP data set. Given the context, say "`data`", and a prefix, say "`m`", we suggest a list of terms (on the left side), each of which corresponds to a list of hit documents (on the right side). This feature has the following advantages. First, it predicts the topic-based query terms which can reflect the user's search intention and thus improve the effectiveness of document retrieval. Second, it supports *autocompletion* which is suggesting query terms instantly as the user types letter by letter. Thus, the user could type less, and find more possible queries as well as their corresponding documents.

There are some challenges to provide such a feature. The first one is the ranking mechanism of terms for increasing the retrieval effectiveness. Since we take into account the topic information, the ranking model is more complicated than the ones used in the conventional methods. In order to calculate the topical coherence of terms in absence of any external information, we make use of the *Latent Dirichlet Allocation* model [14]. LDA is one of the latest *topic models* that can learn hidden topics from text documents in an unsupervised manner, and represent each term as a distribution of those

[1]http://dblp.uni-trier.de/xml/

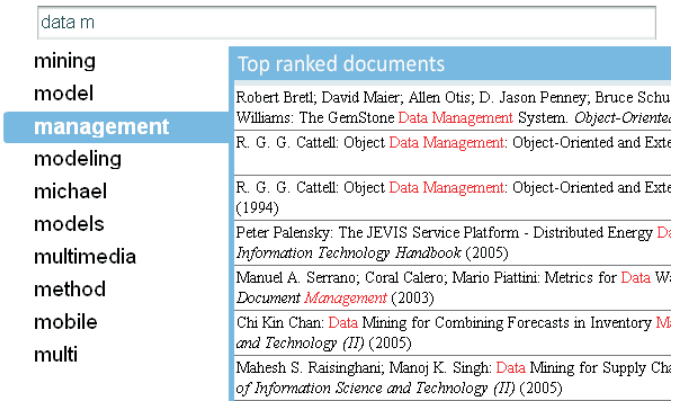


Fig. 1. Results of our approach for the query "`data m`" on the DBLP dataset.

learned topics, which enables the calculation of the topical coherence of a set of terms by choosing the joint probability of them as the relationship measurement. For example, with the support of this model, we can predict that "`transaction`" is more coherent with "`database`" (both of them are strongly related to the topic "database management") than "`traffic`" on the DBLP data set. Moreover, in order to ensure that the suggested terms are more suitable to retrieve the underlying documents, we make use of a language model [15] approach to estimate the likelihood of sampling a term given the documents retrieved by previous query terms.

The second challenge is suggesting terms in real time. Similar works (known as autocompletion and type-ahead search) are extensively studied to improve the efficiency of document ranking when the query consists of incomplete terms (i.e., prefixes) [16], [17], [18], [19]. However, our problem is more complicated than theirs due to two reasons. First, we should not only rank the top-k terms but also their corresponding documents, thereby increasing the complexity. Second, because we incorporate the topics into our ranking model rather than using simple statistics, a sophisticated algorithm should be employed to support fast term suggestion. We use an index structure, *trie*, to efficiently access all possible terms with a particular prefix, and design a progressive algorithm to find the top-k terms based on our proposed model.

Our main contributions are summarized as follows.

- We propose a topic-based query term suggestion model to help users formulate high-quality queries.
- We develop an efficient algorithm to suggest query terms as users type in queries letter by letter, which can save users' typing efforts.
- Extensive experiments on two real data sets show the effectiveness of our topic-based suggestion model and the efficiency of our algorithm.

The rest of the paper is organized as follows. The related works are reviewed in Section II. In Section III, we introduce our generative model and the estimation method. We discuss a progressive algorithm for ranking terms in Section IV and conduct experiments in Section V. Finally, we conclude the paper in Section VI.

II. Related Works

The most related work is Complete Search [20]. It takes the number of documents shared by the suggestions and the context as the ranking mechanism. Bast et al. [9] extends this feature by incorporating the term clusters into the documents.

In general, current techniques used in interactive query expansion can be divided into document-based, term-based and concept-based methods (see [21] for a good survey). Document-based methods suggest the relevant terms in the documents with the support of some statistics (e.g., co-occurrence) [10], [11]. Term-based methods try to discover the relationship of terms from manually constructed thesauruses [6], relevance feedback [6], [7], external dictionaries (e.g., WordNet) [8], and query logs [12]. Concept-based methods map query terms to a number of concepts by means of knowledge bases (e.g., Wikipedia and web directory) [9], [22] and query logs [23].

The topic information has been used to analyze queries in the community of information retrieval. Huang et al. employ LDA to measure the similarity of two queries with their clicked document sets [24]. Wang et al. use Information Bottleneck theory to classify terms into distinct topic clusters, which are used to measure the query term similarity. He et al. [25] use the detected topics of queries to substitute original queries. The above techniques try to discover the topics behind a query for capturing the user's search intention.

Autocompletion allows the user to avoid unnecessary typings, saving not only time but also cognitive burdens [26]. The most popular index structure in autocompletion is *trie* [16], [17], [26]. This tree-based structure compresses terms sharing a common prefix in a corresponding subtree. In addition, HYB [18] and other sophisticated index structures [19] are also designed for fast access.

Our work differs from existing works in the following aspects. First, compared with traditional interactive query expansion techniques, our model takes into account the topic information and learns hidden topics from the underlying documents with the support of LDA. This model is effective to suggest terms that are closely related to the context and that are suitable to retrieve documents. Second, compared with other works employing topic information to measure relationship of queries, our approach focuses on measuring the topical coherence between a term and the context, thereby leading to a better query term ranking mechanism. Third, we extend interactive query expansion by supporting autocompletion and design efficient algorithms to implement it. This extension can further save both the time and cognitive burdens.

III. Model for Ranking Terms

In this section, we propose a model for effectively suggesting a query term. The basic idea of our model is to estimate the *probability* of each term conditioned on the context (i.e., previous query terms) as a ranking score. Intuitively, if a term is more topically coherent with the context and is more likely to be sampled from the retrieved documents, it would obtain a higher probability. Take the DBLP data set as an example.

Given a context, say "database", and a prefix, say "tra", consider a term "transaction". Since "transaction" and "database" are topically coherent and the former occurs frequently in the documents retrieved by the latter, the probability of "transaction" would be higher. Our model focuses on estimating two important probabilities from the documents: the probability of a term conditioned on topics and the probability of sampling a term from a document. Both are used to estimate the score of each term.

- An LDA model [14] is utilized to learn the term distribution over each topic from the underlying documents. Unlike the conventional term clustering techniques used in [9], [23], LDA can be classified as a *soft-clustering* technique which allows a term to appear in multiple clusters (i.e., topics) and takes into account the degree of a term belonging to each topic.
- The term distribution over a set of documents is learnt by using a language model [15]. The language model approach can capture the property of the documents and predict the *likelihood* of sampling a specific term.

Formally, the probability of each terms with a prefix conditioned on the context is defined as follows. Let $s = q_1, q_2, \ldots, q_{|s|}$ be a sequence of previous query terms, that is, the context. Let p be the prefix of the term that the user is typing in and $C = \{c_1, c_2, \ldots, c_{|C|}\}$ be a set of complete terms with the prefix p. We denote $D = \{d_1, d_2, \ldots, d_{|D|}\}$ to be a collection of documents retrieved by the context s and $T = \{t_1, t_2, \ldots, t_{|T|}\}$ to be a set of topics. The task of our model is to estimate the probability $P(c|s)$ for each $c \in C$ and suggest the top terms with highest probabilities.

We propose a generative model to estimate the probability $P(c|s)$. Each topic t_i in the topic set T and each document d_j in the document collection D are incorporated into the probability estimation (See Equation (1)).

$$P(c|s) = \sum_{t_i \in T} \sum_{d_j \in D} P(c|s; t_i, d_j) \cdot P(t_i, d_j|s) \tag{1}$$

Intuitively, given a context s, our model would examine the likelihood that a user issues the term c if t_i is the topic of c and d_j is an expected answer. It is conceptually similar to the query likelihood $P(q|d_i)$ in the language model approaches [15]: we first estimate a model from each topic and each document, and then compute the likelihood of generating c according to this model as well as the context s. To estimate $P(c|s; t_i, d_j)$, ideally we should enumerate every possible term which is not only under the topic t_i, but also used by users who want to retrieve document d_j given a context s. However, in reality it is very difficult to achieve this goal. So we assume that c is conditional independent of s given t_i and d_j and use $\lambda P(c|t_i)+(1-\lambda)P(c|d_j)$ (where λ is a tuning parameter) as an estimate for $P(c|s; t_i, d_j)$ (Similar methods are used in [27], [28]). On the other hand, the probability $P(t_i, d_j|s)$ can be analogous to the *prior* of choosing t_i and d_j given the context s. The more the topic t_i and the document d_j are relevant to the context s, the more they have the *ability* to generate any term. We assume that t_i and d_j are independent of each other, thereby resulting in $P(t_i, d_j|s) = P(t_i|s) \cdot P(d_j|s)$. Hence, $P(c|s)$ is further estimated by Equation (2).

$$\begin{aligned} P(c|s) &= \sum_{t_i \in T} \sum_{d_j \in D} \big(\lambda P(c|t_i) + (1-\lambda) P(c|d_j)\big) P(t_i|s) P(d_j|s) \\ &= \lambda \sum_{t_i \in T} P(c|t_i) P(t_i|s) + (1-\lambda) \sum_{d_j \in D} P(c|d_j) P(d_j|s) \end{aligned} \tag{2}$$

Clearly, Equation (2) shows that the probability $P(c|s)$ depends on two factors. The first factor is the relationship between c and s in terms of topics. The second one is the likelihood of sampling c from each document d_j retrieved by s. The two factors are combined linearly with a parameter λ. Note that in the case that the context is an empty string (i.e., $s = \varepsilon$), $P(c|s)$ is simply estimated by the document frequency of c.

Now, we discuss the computation of $P(c|s)$ in Equation (2). $P(c|t_i)$ is trained from the entire document collection via LDA, and $P(t_i|s)$ can be rewritten to Equation (3) with Bayes formula.

$$P(t_i|s) = \frac{P(s|t_i) \cdot P(t_i)}{P(s)} \tag{3}$$

Because the assumption that terms are independent of each other conditioned on topics is used in LDA and $s = q_1, q_2, \ldots, q_{|s|}$, we get Equation (4) (where $s' = q_1, q_2, \ldots, q_{|s|-1}$).

$$P(t_i|s) = \frac{P(t_i) \cdot \prod_{l=1}^{|s|} P(q_l|t_i)}{P(s') \cdot \sum_{t_j \in T} P(q_{|s|}|t_j; s') \cdot P(t_j|s')} \tag{4}$$

Obviously, both $P(s')$ and $P(q_{|s|}|t_j; s') \cdot P(t_j|s')$ have been computed from the estimate of $P(q_{|s|}|q_1, q_2, \ldots, q_{|s|-1})$ in the previous round.

The second factor of $P(c|s)$ is a sum of likelihoods of generating c from each document, i.e., $\sum_{d_j \in D} P(c|d_j) \cdot P(d_j|s)$. We use the Jelinek-Mercer smoothing technique [15] in language model approach to estimate $P(c|d_j)$ in Equation (5).

$$P(c|d_j) = (1-\gamma)\frac{\mathsf{count(c, d_j)}}{\mid d_j \mid} + \gamma \frac{\mathsf{count(c, D)}}{\mid D \mid} \tag{5}$$

where $\frac{\mathsf{count(c,d_j)}}{|d_j|}$ is term frequency of c in the document d_j and $\frac{\mathsf{count(c,D)}}{|D|}$ the frequency of c in a *background* model, i.e, the entire collection of documents, D. In addition, γ is a tuning parameter. The intuition of Equation (5) is that we interpolate the maximum likelihood estimation, $\frac{\mathsf{count(c,d_j)}}{|d_j|}$, with the background model by using a tuning parameter γ. This smoothing technique has been intensively examined and shown effective [15]. $P(d_j|s)$ is the prior probability that the document d_j generates any term given the context s. We use a normalized ranking score of d_j under s to estimate it (the TF-IDF model is used in our experiments).

Fig. 2 gives an example to illustrate our model. A sample document set from the DBLP data set is provided in Fig. 2(a). Note that we only consider the paper titles for ease

ID	Sample Documents in DBLP dataset (only paper titles)
0	Benchmarking Decision Models for Database Management Systems
1	A Dynamic Data Model for a Video Database Management System
2	Using Economic Models to Allocate Resources in Database Management Systems
3	Some Models of a Distributed Database Management System with Data Replication
4	A General Model for Event Specification in Active Database Management Systems
5	Mining Protein Database using Machine Learning Techniques
6	Mining Sequential Patterns across Multiple Sequence Databases
7	Data Mining: Machine Learning, Statistics, and Databases
8	Declarative Database Management in SQLServer
9	Machine Learning and Data Mining

(a) A sample document set.

Term	Topic			P(term)	DF
	t_0	t_1	t_2		
database	0.050	0.010	0.008	0.023	9
data	0.012	0.013	0.047	0.024	4
decision	0.004	0.015	0.038	0.019	1
model	0.060	0.020	0.001	0.027	5
management	0.003	0.004	0.040	0.016	6
mining	0.020	0.030	0.010	0.020	4
machine	0.010	0.040	0.005	0.018	3
system	0.040	0.020	0.005	0.022	5
statistic	0.002	0.036	0.015	0.018	1
sequence	0.001	0.045	0.018	0.021	1

(b) Probabilities of terms under topics and DFs.

$P(c\|t_0) = 0.06$	$P(c\|d_0)P(d_0\|s) = 1/54$
$P(t_0\|s) = \frac{P(s\|t_0)\cdot P(t_0)}{P(s)} = 0.7246$	$P(c\|d_1)\cdot P(d_1\|s) = 1/63$
$P(c\|t_1) = 0.02$	$P(c\|d_2)\cdot P(d_2\|s) = 1/72$
$P(t_1\|s) = \frac{P(s\|t_1)\cdot P(t_1)}{P(s)} = 0.1449$	$P(c\|d_3)\cdot P(d_3\|s) = 1/63$
$P(c\|t_2) = 0.001$	$P(c\|d_4)\cdot P(d_4\|s) = 1/72$
$P(t_2\|s) = \frac{P(s\|t_2)\cdot P(t_2)}{P(s)} = 0.1159$	$P(c\|d_i)\cdot P(d_i\|s) = 0\ (i = 5, 6, 7, 8)$
Result:	$P(c\|s) = 0.5\cdot\sum_{i=0}^{2} P(c\|t_i)P(t_i\|s) + 0.5\cdot\sum_{j=0}^{8} P(c\|d_j)P(d_j\|s) = 0.0623$

(c) Computation of $P(\texttt{model}|\texttt{database})$.

Fig. 2. An example document collection on the DBLP data set.

of presentation. In addition, a subset of term probabilities under topics and document frequencies (DF) are listed in Fig. 2(b) (stop words are eliminated). Note that $P(term) = \sum_{t_i \in T} P(term|t_i)P(t_i)$ and $P(t_i) = \frac{1}{|T|} = \frac{1}{3}$.

For example, suppose a user types in a prefix "d". The terms are ranked according to their document frequencies (see Figure 2(b)) due to the empty context: {database, data, decision, ...}. Consider a chosen term "database" which retrieves a set of documents with the TF-IDF model. Then the user types in a prefix "m", we compute the probability $P(c|\texttt{database})$ for each candidate term according to Equation (2), (3), (4) and (5) by assuming that $\lambda = 0.5$ and $\gamma = 0$. Figure 2(c) provides the computation of $P(\texttt{model}|\texttt{database})$ where c is "model" and s is "database". The other probabilities can be computed in the same manner. Thus we rank the terms as follows: $\{\texttt{model}(0.0623), \texttt{management}(0.0566), \texttt{mining}(0.0365), \ldots\}$. This ranking list is different with the one based on co-occurrence where "management" would be ranked higher than "model". Similarly, after choosing a term, say "model", other terms would be further suggested.

IV. Progressive Ranking Algorithm

In this section, we study how to efficiently suggest a query term. A straightforward way to suggest terms is to first estimate the probabilities discussed in Section III for every term and sort them in descending order. However, because it enumerates all candidates before applying a sort algorithm, this method is inefficient, especially for large numbers of candidates. In contrast, we design a progressive ranking algorithm to efficiently find the top-k terms for a prefix rather than enumerating all candidates. In this section, we firstly introduce the index structure, trie, and then discuss a threshold-based ranking algorithm.

The Trie Index: In order to efficiently search terms with a particular prefix, we introduce *trie* to index all terms in the entire collection of documents. Trie is a tree structure where each path from the root to a leaf node corresponds to a term. Given a prefix, we can find its corresponding node, and traverse all its leaf nodes to obtain the corresponding terms. Fig. 3 provides a trie of seven sample terms. The term "model" has a trie node 10 and its prefix "mod" has a node 8. Consider a prefix

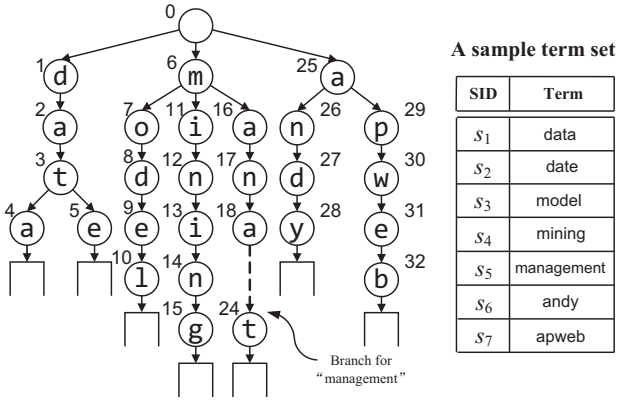


Fig. 3. Trie index of a sample term set.

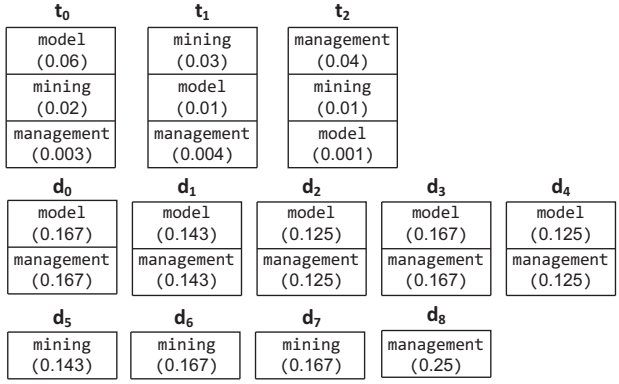


Fig. 4. An example of our ranking algorithm on the DBLP data set.

"m" corresponding to node 6, the terms with this prefix are nodes 10 ("model"), 15 ("mining") and 24 ("management"). In each leaf node, we maintain an inverted list which stores the identifiers of all documents containing the corresponding terms, which are sorted by the TF-IDF scores. With the support of the inverted lists, we can rank the documents efficiently. See [16] for details. For example, suppose that the query terms are "model management". The corresponding nodes of the two terms are 10 and 24 respectively. Both have an inverted list which contains a subset of documents in Fig. 2(a). Thus, the intersect of the two lists are documents containing both "model" and "management".

The Ranking Algorithm: Recall that the estimation of the probability $P(c|s)$ (see Equation (2)) is a monotonous function (as either $P(c|t_i)$ or $P(c|d_j)$ increases, $P(c|s)$ would increase accordingly). In this case, we can apply a threshold algorithm (TA) [29] to find the top-k answers efficiently instead of computing $P(c|s)$ for every term. The algorithm scans the multiple lists that present the different rankings of the same set of terms in related topics and documents. An upper bound T is maintained for the overall scores of the unseen terms, and is computed by applying the ranking function to the partial score of the last seen term in every list. When a new term becomes seen, we update T in order to represent the new upper bound of unseen ones. When scanning a particular term, say c, in a list, we obtain its partial scores in other lists by means of two forward indices: $\mathsf{FWD_t}$ and $\mathsf{FWD_d}$ that respectively maps a topic and a document to the corresponding terms. If the aggregation of such partial scores is greater than or equal to the current T, it means that the score of c is greater than any unseen term. If the size of the result set R is smaller than k, c can be safely inserted into it; otherwise, we can terminate the algorithm. We use a heap structure to maintain the order of the terms in R.

In detail, when the user issues a query consisting of a context s and a prefix p, the algorithm finds the top-k answers as follows. We find the list of terms in each topic t_i by means of $\mathsf{FWD_t}$ and use the trie to select the terms with the prefix p. This list is sorted in descending order of $P(c|t_i)$. Similarly, for each document d_j in D, we can prepare the list of terms that are sorted in descending order of $P(c|d_j)$ with the support of $\mathsf{FWD_d}$. Note that the documents in D are top ranked documents retrieved by s and the ones with insignificant $P(d_j|s)$ are omitted for the purpose of efficiency in the implementation. After obtaining the multiple lists, we use the threshold algorithm to scan all of them and find the top-k terms progressively. The complexity of the algorithm depends on the number of steps used to scan the lists.

Fig. 4 gives an example to illustrate our algorithm, where the context s is "`database`" and the prefix is "`m`". For each topic and document, we prepare lists of terms whose corresponding nodes is the children of node `6` in the trie. The terms in each list is sorted by either $P(c|t_i)$ or $P(c|d_j)$ (the scores in parentheses). Consider the topic t_0. It contains three candidate terms whose scores are 0.06, 0.02 and 0.003 respectively. Moreover, the parameters in Equation (2) (i.e., λ, $P(t_i|s)$ and $P(d_j|s)$) are also given as the inputs of the algorithm (see Fig. 2(c)). When applying the threshold algorithm to find the top-1 term, we scan all lists. At the first step, the seen terms are the first object in each list. We firstly make use of Equation (2) to compute T, and the result is 0.107. Then we compute the scores of the seen terms: $P($`model`$|$`database`$) =$ `0.0623`, $P($`mining`$|$`database`$) =$ `0.0566` and $P($`management`$|$`database`$) =$ `0.0365`, followed by maintaining them in a heap. Since there is no score greater than T, we continue to scan the list and update T to 0.049 by aggregating the partial scores of the second objects in lists. Since $0.0623 > 0.049$, "`model`" can be safely inserted into R and the algorithm terminates.

V. Experiments

In this section, we conducted extensive experimental studies and evaluated the effectiveness and efficiency of our proposed method on two real data sets.

Data sets: 1) DBLP[2]: a bibliography corpus in computer science, where each document contains title, year, authors and venue of a paper. 2) DBLife[3]: various kinds of web pages in the community of database research, e.g., the homepages of researchers, tour information of a city, personal blogs and photos, etc.. The DBLife data set is noisier than the DBLP data set and is close to Web data. Table I provides the statistics of the two data sets, where *len* is the number of unique terms in a document and Σ is the set of unique terms in the entire collection of documents. Note that both data sets were preprocessed by eliminating stop words.

TABLE I
Dataset statistics

Data Sets	# of doc.	*avg_len*	*max_len*	*min_len*	$\lvert\Sigma\rvert$
DBLP	1109727	13.5	134	1	406015
DBLife	8301	224	7057	1	109716

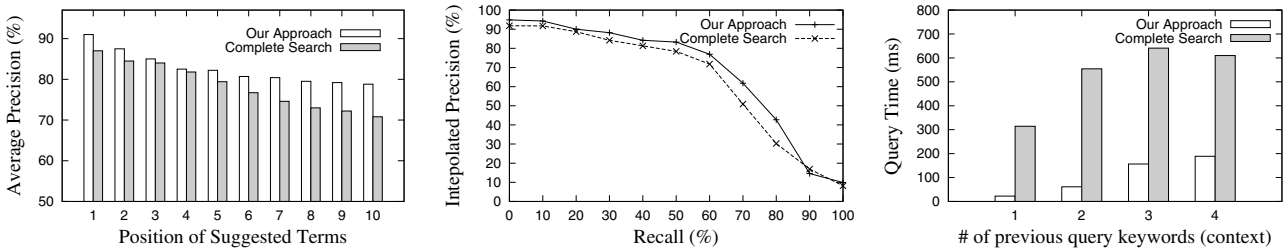
Query Sets: 1) For DBLP data set, we selected 100 frequently issued queries from the query logs of two real search engines on DBLP: iSearch[4] and DBLPSearch[5]. Each query is composed of a context and a prefix. For example, a query "`graph m`" with length of 2 has a context "`graph`" and a prefix "`m`". The average length of the queries is 2.58. 2) For DBLife data set, since we cannot obtain query logs, we selected queries related to database research from the logs from iSearch and DBLPSearch. The average length of queries is 2.8.

Experiment Settings: We implemented our approach and the baseline, "word refinement" used in Complete Search [20]. The baseline method examines all possible terms and takes the number of documents which are shared by the context and each term (i.e., co-occurrence) as the ranking score. All the algorithms were implemented in C++ and compiled using GCC 4.2.3 with -O3 flag. All the experiments were run on a Ubuntu Linux machine with an Intel Core 2 Quad X5450 3.00GHz processor and 4 GB memory. The parameters of our algorithm were set as follows:

- $k = 10$, $\lambda = 0.3$ and $\gamma = 0.1$ for the DBLP data set.
- $k = 10$, $\lambda = 0.6$ and $\gamma = 0.1$ for the DBLife data set.

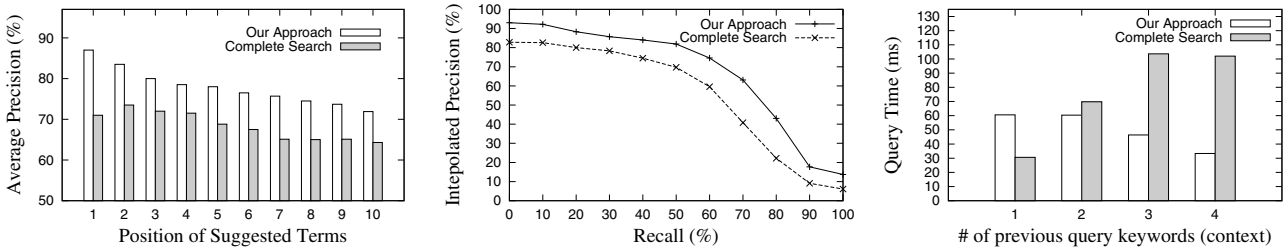
Evaluation Metrics: We asked 10 volunteers, who are familiar with the underlying data sets, to test the two algorithms. They labeled the relevance of each suggested query term in a *blind test* manner, that is, they did not know which algorithm returned the terms. We qualified the effectiveness through standard measures of precision and recall[6], and employed two

[2]http://dblp.uni-trier.de/xml/
[3]http://dblife.cs.wisc.edu/download/surfaceWeb-2007-07-25.tgz
[4]http://tastier.cs.tsinghua.edu.cn/dblpsearch/
[5]http://dblp.ics.uci.edu/
[6]We use all relevant terms from the two approaches to compute recall.



(a) Average precision at position n. (b) Precision-recall curves. (c) Efficiency at various context lengths.

Fig. 5. Performance comparisons on the DBLP data set.



(a) Average precision at position n. (b) Precision-recall curves. (c) Efficiency at various context lengths.

Fig. 6. Performance comparisons on the DBLife data set.

aggregate metrics: average precision at position n ($P@n$) and precision-recall curves (see Section V-A). We compared the efficiency and the scalability in Section V-B and V-C.

A. Evaluation of Effectiveness

We first evaluate the effectiveness of the two methods on the DBLP data set. Observed from Figs. 5(a) and 5(b), our method achieves higher average precision at each position of suggested terms, and yields better effectiveness at nearly every percentage of recall. Figs. 6(a) and 6(b) provide the results of the DBLife data set. Our method is significantly better than the baseline. For example, the gains in $P@1$ were of 87% for our approach and of 71% for Complete Search. In addition, the gap between our approach and the baseline method is larger on the DBLife data set than that on the DBLP data set. It shows that our approach is more capable to handle noisy data and outperforms the baseline method.

TABLE II
RESULTS FOR TWO SAMPLE QUERIES ($k = 5$).

Queries	Our Approach	Complete Search
DBLife data set		
semantic o	ontology	org
	ontologies	open
	org	order
	overview	object
	object	oriented
DBLP data set		
real time s	system	system
	scheduling	**symposium**
	software	scheduling
	simulation	software
	symposium	simulation

Table II shows the results for two sample queries, where top-5 terms are displayed. Observed from the results on the DBLife data set, our approach suggests the more related terms (e.g., "`ontology`"), while most results of Complete Search have little relationship with "`semantic`". Moreover, our approach provides a better term ranking. When a query "`real time s`" was issued on the DBLP data set, a non-specific term, "`symposium`" ranked second in the result list of Complete Search, because many papers whose titles contain "`real time`" are published in some kinds of *symposium*. However, this highly-frequent term is not a good choice to specify the user's information need due to its poor *term discrimination* [30]. In contrast, our approach ranked the three other terms ("`scheduling`", "`software`" and "`simulation`") ahead of "`symposium`", which are more related to the context and more possible to hit relevant documents.

Our method outperforms the baseline because it discovers the topical coherence of terms with the support of LDA. The topical coherence can be used to describe the relationship of terms in a query, capturing the user's information need. In addition, since we also take into account the possibility of sampling a term from documents, it guarantees that the suggested terms are suitable to hit relevant documents.

B. Evaluation of Efficiency

We examined the efficiency of our progressive ranking algorithm discussed in IV. The experiment results on the two data sets are shown in Figs. 5(c) and 6(c) respectively. The results show that our algorithm achieves better search efficiency.

- For the DBLP data set, our algorithm achieves better efficiency than the method of counting co-occurrences used by Complete Search. For example, when the number of context keywords is 1 or 2, our algorithm outperforms the baseline by an order of magnitude.

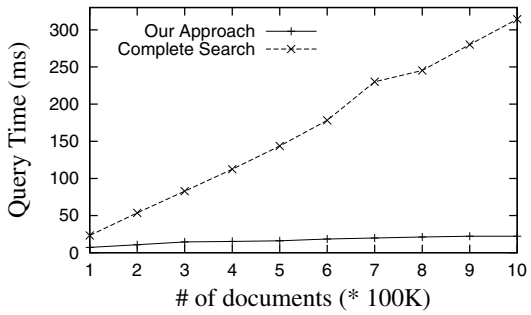


Fig. 7. Scalability on the DBLP data set (# of context = 2).

- For the DBLife data set, the baseline algorithm is fast in the situation of one context keyword. The reason is that the number of documents is small (8301), which means that counting the shared documents of two terms is very easy. Unfortunately, as the increase of the number of context terms, the time increases significantly. In contrast, the time of our algorithm is stable, or even decreases, because the number of candidate lists (see Section IV) is stable or smaller as the number of context terms increases.

Our ranking algorithm is more efficient than the baseline due to two reasons. First, we only take terms in the top ranked documents retrieved by the context as candidates, which reduces the size of candidate set. Second, we employ a threshold algorithm to rank the top-k results rather than computing all possible scores (as used in the Complete Search).

C. Evaluation of Scalability

In this section, we compare the scalability of the two algorithms on the DBLP data set. Initially, the entire document set was empty, and we inserted 100K documents at each time. We run the two algorithms at each scope of the document set. Fig. 7 shows the experiment results. We observe that our algorithm scales better than the baseline. As the increase of the number of documents, the query time of our algorithm increases more smoothly. For example, the query time of our algorithm increases from 7.2ms to 22.2ms, while the one of Complete Search increases from 23.2ms to 314.4ms.

VI. Conclusion

In this paper, we have studied the problem of interactive query term suggestion for improving the performance of information retrieval. We introduced topic information into our probabilistic ranking model to suggest terms that are not only topically coherent to the initial query and are suitable to retrieve the underlying documents. We proposed the estimation and calculation of the ranking model. In order to suggest terms with supporting autocompletion, we utilized the trie structure to access terms with a prefix and devised a progressive ranking algorithm to find the top-k terms efficiently. We have implemented our approach and examined it on two real data sets. The experimental results show that our approach achieves high performance and outperforms the baseline in both effectiveness and efficiency.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under grant No. 60833003. We appreciate the helpful comments of the anonymous reviewers.

References

[1] Y. Nemeth, B. Shapira, and M. Taieb-Maimon, "Evaluation of the real and perceived value of automatic and interactive query expansion," in *SIGIR*, 2004, pp. 526–527.
[2] S. Cronen-Townsend, Y. Zhou, and W. B. Croft, "Predicting query performance," in *SIGIR*, 2002, pp. 299–306.
[3] Y. Zhou and W. B. Croft, "Query performance prediction in web search environments," in *SIGIR*, 2007, pp. 543–550.
[4] M. Magennis and C. J. van Rijsbergen, "The potential and actual effectiveness of interactive query expansion," in *SIGIR*, 1997, pp. 324–332.
[5] H. Joho, M. Sanderson, and M. Beaulieu, "A study of user interaction with a concept-based interactive query expansion support tool," in *ECIR*, 2004, pp. 42–56.
[6] D. Harman, "Towards interactive query expansion," in *SIGIR*, 1988, pp. 321–331.
[7] D. Harman, "Relevance feedback revisited," in *SIGIR*, 1992, pp. 1–10.
[8] Z. Gong, C. W. Cheang, and L. H. U, "Web query expansion by wordnet," in *DEXA*, 2005, pp. 166–175.
[9] H. Bast, D. Majumdar, and I. Weber, "Efficient interactive query expansion with complete search," in *CIKM*, 2007, pp. 857–860.
[10] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *SIGIR*, 1996, pp. 4–11.
[11] R. Kraft and J. Y. Zien, "Mining anchor text for query refinement," in *WWW*, 2004, pp. 666–674.
[12] Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, E.-P. Lim, and H. Li, "Web query recommendation via sequential query prediction," in *ICDE*, 2009, pp. 1443–1454.
[13] D. Kelly, K. Gyllstrom, and E. W. Bailey, "A comparison of query and term suggestion features for interactive searching," in *SIGIR*, 2009, pp. 371–378.
[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," in *NIPS*, 2001, pp. 601–608.
[15] C. Zhai, "Statistical language models for information retrieval: A critical review," *Foundations and Trends in Information Retrieval*, vol. 2, no. 3, pp. 137–213, 2008.
[16] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *WWW*, 2009, pp. 371–380.
[17] G. Li, S. Ji, C. Li, and J. Feng, "Efficient type-ahead search on relational data: a tastier approach," in *SIGMOD*, 2009, pp. 695–706.
[18] H. Bast and I. Weber, "Type less, find more: fast autocompletion search with a succinct index," in *SIGIR*, 2006, pp. 364–371.
[19] H. E. Williams, J. Zobel, and D. Bahle, "Fast phrase querying with combined indexes," *ACM TOIS*, vol. 22, no. 4, pp. 573–594, 2004.
[20] Complete search. Max-Planck-Institute for Informatics. [Online]. Available: http://dblp.mpi-inf.mpg.de/dblp-mirror/index.php
[21] B. Billerbeck, "Efficient query expansion," Ph.D. dissertation, RMIT University, Melbourne, Australia, 2005.
[22] Y. Chen, G.-R. Xue, and Y. Yu, "Advertising keyword suggestion based on concept hierarchy," in *WSDM*, 2008, pp. 251–260.
[23] B. M. Fonseca, P. B. Golgher, B. Pôssas, B. A. Ribeiro-Neto, and N. Ziviani, "Concept-based interactive query expansion," in *CIKM*, 2005, pp. 696–703.
[24] S. Huang, Q. Zhao, P. Mitra, and C. L. Giles, "Hierarchical location and topic based query expansion," in *AAAI*, 2008, pp. 1150–1155.
[25] X. He, J. Yan, J. Ma, N. Liu, and Z. Chen, "Query topic detection for reformulation," in *WWW*, 2007, pp. 1187–1188.
[26] A. Nandi and H. V. Jagadish, "Effective phrase prediction," in *VLDB*, 2007, pp. 219–230.
[27] O. Kurland and L. Lee, "Corpus structure, language models, and ad hoc information retrieval," in *SIGIR*, 2004, pp. 194–201.
[28] O. Kurland, "The opposite of smoothing: a language model approach to ranking query-specific document clusters," in *SIGIR*, 2008, pp. 171–178.
[29] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *PODS*, 2001.
[30] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," pp. 273–280, 1997.