# Assessment of Learning to Rank Methods for Query Expansion

**Bo Xu, Hongfei Lin, and Yuan Lin**
*Department of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China.*
*E-mail: xubo2011@mail.dlut.edu.cn; hflin@dlut.edu.cn; zhlin@dlut.edu.cn*

**Pseudo relevance feedback, as an effective query expansion method, can significantly improve information retrieval performance. However, the method may negatively impact the retrieval performance when some irrelevant terms are used in the expanded query. Therefore, it is necessary to refine the expansion terms. Learning to rank methods have proven effective in information retrieval to solve ranking problems by ranking the most relevant documents at the top of the returned list, but few attempts have been made to employ learning to rank methods for term refinement in pseudo relevance feedback. This article proposes a novel framework to explore the feasibility of using learning to rank to optimize pseudo relevance feedback by means of reranking the candidate expansion terms. We investigate some learning approaches to choose the candidate terms and introduce some state-of-the-art learning to rank methods to refine the expansion terms. In addition, we propose two term labeling strategies and examine the usefulness of various term features to optimize the framework. Experimental results with three TREC collections show that our framework can effectively improve retrieval performance.**

## Introduction

When a user submits a query to a search engine, he may fail to obtain what he desires because the query is ambiguous and difficult for the search engine to understand clearly. To interpret the query better, query refinement methods have been developed to reformulate the query by incorporating or removing some query terms, thus improving the retrieval performance, which can be divided into two categories based on the query length. For a long query, query refinement methods have usually been designed to remove some irrelevant or redundant query terms; for a short query, the methods are developed to add some relevant terms to expand the original query. Meanwhile, to reformulate the query better, query terms usually acquire different weights based on their degree of relevance to the main idea of the original query. Here our research is focused mainly on the query expansion methods.

Pseudo relevance feedback (PRF), one of the query expansion methods, obtains its expansion terms from top-ranked documents of the initial retrieval and has been widely used in information retrieval (IR; Cao, Nie, Gao, & Robertson, 2008; Lee, Croft, & Allan, 2008; Lv, Zhai, & Chen, 2011; Rocchio, 1971; Tao & Zhai, 2006; Xu & Croft, 1996). However, some of the expansion terms used in PRF may negatively impact the retrieval performance because they tend to be irrelevant to the original query. Therefore, it is necessary to exclude the irrelevant terms and to reformulate the query better.

To refine the expansion terms, some approaches are needed to distinguish the relevant terms from the irrelevant ones among a large set of candidate expansion terms. In IR, learning to rank methods solve ranking problems by employing machine learning techniques and have been proved powerful for improving retrieval performance (Liu, 2009; Qin, Liu, Xu, & Li, 2010). In fact, ranking, as the central issue of IR, aims to generate a list of documents based on the degree of relevance to a given query. In the list, a document with a higher relevance degree is ranked before the one with a lower relevance degree. Therefore, learning to rank methods, as effective ranking algorithms, can help to distinguish relevant documents from irrelevant ones. However, few studies have attempted to apply learning to rank methods to the term refinement for query expansion. Given that, we investigate whether learning to rank methods can help in choosing more useful expansion terms.

We propose a query expansion framework based on PRF, in which learning to rank methods are introduced to refine

Received June 21, 2014; revised November 4, 2014; accepted November 12, 2014

 DOI: 10.1002/asi.23476

the expansion terms. First, given the original query, we conduct initial retrieval to obtain some top-ranked documents. Then, from these documents, we choose a large set of frequently appearing terms as our candidate expansion terms. Next, we refine the terms using learning to rank methods. Finally, after the refinement, more useful terms are chosen to reformulate the query, and results are retrieved again.

The contributions of this work are as follows: (a) to present the usefulness of expansion terms, we propose two term truth labeling strategies; (b) we define a large set of term features and examine their effectiveness based on their different combinations; and (c) we examine and compare several state-of-the-art learning to rank methods for term refinement.

## Related Work

Query refinement methods have been studied for years and can be divided into two categories based on the query length. For a long query, methods have been developed to remove or reduce the weights of some redundant query terms. For example, Jones and Fain (2003) attempted to predict and remove some useless query terms based on query logs. Kumaran and Allan (2008a, 2008b) tried to find optimal subqueries by using maximum spanning tree weighted by mutual information. For a short query, methods have been developed to expand the query with some query-related terms. Query expansion methods, especially PRF, have been widely used in IR (Buckley, 1994; Cronen-Townsend, Zhou, & Croft, 2004; Metzler & Croft, 2007; Xu & Croft, 1996). Traditional PRF has been implemented in retrieval models such as the vector space model (Rocchio, 1971), probabilistic model (Robertson, Walker, Beaulieu, Gatford, & Payne, 1996), relevance model (Lavrenko & Croft, 2001), and mixture model (Zhai & Lafferty, 2001). Some previous studies have attempted to improve PRF in terms of document refinement. For example, Lee et al. (2008) proposed a clustered-based resampling method to generate pseudo relevance documents, where dominant documents for a topic can be sampled. Lv et al. (2011) developed a boosting approach, FeedbackBoost, to accommodate many basic feedback methods using a boosting approach. Unlike these studies, our method improves the effectiveness of PRF by refining expansion terms.

Because different terms play different roles in a query, some studies have been focused on term weighting. Lease, Allan, and Croft (2009) developed a supervised learning method, Regression Rank, for term weighting in descriptive queries. Similarly, they introduced an improved Markov random field model to deal with verbose queries (Lease, 2009). Although their methods are not designed to expand the query, they inspired us to select expansion terms in a supervised way. Lee, Chen, Kao, and Cheng (2009) introduced a rich set of linguistic and statistical features to explore the underlying relationship among query terms. Cao et al. (2008) introduced a classification method to select good expansion terms from a large set of terms. In comparison with their methods, we adopt ranking methods to choose the expansion terms. Ranking methods can explicitly get the sorted term list to select the most useful terms directly, so they may be a better choice for refining the terms.

Many tasks in IR and text mining can be solved by using ranking methods. For example, question answering systems answer a question by ranking many candidate answers (Surdeanu, Ciaramita, & Zaragoza, 2008), and recommender systems provide users with appropriate items by ranking the preference information (Sun, Wang, Gao, & Ma, 2012). To solve the ranking problem, learning to rank methods have been proposed and proved effective to improve retrieval performance (Burges, 2010; Burges, Ragno, & Le, 2006; Cao et al., 2006; Freund, Lyer, Schapire, & Singer, 2003; Joachims, 2002; Wu, Burges, Svore, & Gao, 2008; Xu & Li, 2007).

Some studies have optimized the process of query expansion by using ranking methods. Lin, Lin, Jin, and Ye (2011) exploited social annotations as the source of expansion terms for enriching the query. Similarly, Oliveira et al. (2012) introduced unsupervised tag recommendation methods to expand entity-related queries from Wikipedia articles. Our framework has two main differences when compared with these. The first is the source of expansion terms. Instead of obtaining expansion terms from external sources, we obtain our candidate expansion terms from the top-ranked documents, namely, the feedback documents, focusing on the improvement of PRF. The second difference is that we compare various state-of-the-art learning to rank methods for query expansion, seeking to develop better term ranking models.

## Methods

This section formulates our query expansion framework in detail. As mentioned above, top-ranked documents of initial retrieval are used as the source of candidate expansion terms in our framework, which contains many useful terms that can help in interpretinjg the query. First, we introduce a term-dependence method to choose candidate terms from these documents. Second, we measure the relevance degree of each term to the original query and represent the terms as feature vectors for supervised learning. Finally, we rank terms based on learning to rank methods and choose the final expansion terms from the top of the term ranking list.

### *Candidate Term Selection*

Before presenting our term ranking method for expansion term refinement, we first introduce a method to choose candidate expansion terms from top-ranked documents, namely, the feedback documents. Inspired by the work of Lin et al. (2011), which develops a term-dependence method to choose a set of expansion terms from social annotation resources and achieves relatively high retrieval performance,


DOI: 10.1002/asi

we introduce the term-dependence method to choose candidate terms from feedback documents as the first step.

The term-dependence method leverages the co-occurrence information of the query and an expansion term to measure the importance and relevance of each term and then chooses the terms with the highest dependence scores as candidate terms. Two dependence situations are taken into consideration while measuring term relevance: term co-occurrence with one query term and with two neighboring query terms. They are denoted as the full independence (FI) method and the sequential dependence (SD) method. The final term-dependence (TD) method is the linear interpolation of the two methods (Lin et al., 2011).

In the first presentation of the method, terms are obtained from a social annotation collection, which is a manually edited thesaurus and provides a wealth of information about relevant terms. Instead of using an annotation collection, we use the feedback documents as the source of candidate terms. Statistics given by Lin et al. (2011) show that both the proportion of good terms and the proportion of bad terms are increased by using the social annotation information compared with feedback documents, which indicates that, although more potential good terms may be selected for query expansion, the potential of choosing bad terms for expansion is also increased; that is to say, it may be more difficult to discriminate the good from the bad terms in further selection. Therefore, we obtain the candidate expansion terms from feedback documents. For one thing, a search engine has easier access to feedback documents during the retrieval process, especially for query expansion. For another, the decrease of the proportion of bad terms in feedback documents could reduce the degree of difficulty in further selection and facilitate term refinement, so we apply the TD method to feedback documents.

### *Term Labeling Strategies*

To rank terms in a supervised way, we should label terms as relevant or not, so that in the training process we can generalize information from past queries to predict new ones. We propose two strategies to form a ground truth label for each term.

Intuitively, good expansion terms can improve retrieval performance when added to the original query, and bad terms can hurt the retrieval performance. Based on this idea, our first labeling strategy is as follows. For a given query and a candidate term with respect to the query, we measure the usefulness of the term based on the mean average precision (MAP) change when adding the term to the query. Specifically, a term is labeled relevant when the value of MAP increases after adding it to the original query and irrelevant when the value of MAP decreases. We label a term 1 or 0, indicating relevance or irrelevance.

Because binary relevance degree has limited capability in expressing term relevance for ranking, we introduce some measures to form multiple relevance degrees for each term. To measure the relevance further, we take the number of terms used for query expansion into account when labeling a term because our term ranking goal is to find the most $k$ relevant terms. Table 1 presents the labeling strategy, where $k$ is the number of terms we choose for query expansion.

TABLE 1. Truth table of labeling on terms.

| $label(t)$ | $MAP(q \cup t) - MAP(q) \geq 0$ | $rank(t) \leq k$ |
|---|---|---|
| 2 | True | True |
| 1 | True | False |
| 1 | False | True |
| 0 | False | False |

TABLE 2. Notations for term features.

| Parameter | Description |
|---|---|
| $T = t_1, \ldots, t_m, i$ | A set of candidate terms, size $m$, indexed by $i$ |
| $Q = q_1, \ldots, q_n, j$ | Query $Q$ of length $n$, indexed by $j$ |
| $pair(Q), k$ | Query term pairs in query $Q$, size $l$, indexed by $k$ |
| $C, N$ | The whole collection $C$ containing $N$ documents |
| $S, M$ | The feedback collection $S$ containing $M$ documents |

Terms are sorted by the change of MAP values and $rank(t)$, where $rank(t)$ is the position of term $t$ in the sorted list. Label 2 implies that the term is definitely relevant to the query, label 1 implies that it is possibly relevant, and label 0 implies that it is irrelevant. Ideally, the top $k$ terms in the list should be selected for query expansion.

In our experiment, we tune the parameter $k$ based on the retrieval performance of the original PRF method and choose the optimal value when the performance is highest. Meanwhile, the value of $k$ is also used as the number of terms chosen from the ranking list of candidate terms in the expanded queries.

### *Term Ranking Features*

Unlike traditional application of learning to rank methods, terms are represented here instead of documents as feature vectors to train a term ranking model. Therefore, we define some candidate term features in this section. Term features refer to the statistical information about the term in a document collection. Intuitively, term features should not only distinguish the candidate terms from each other based on the term distribution information but also imply term relevance to the query so that ranking methods can work more efficiently. Based on the idea, some term features are defined and their usefulness is investigated. Some notations and definitions of features are listed in Tables 2 and 3.

To measure the importance of terms in the whole collection, we introduce some classic statistical information in IR as term features for learning to rank (Qin et al., 2010). Term


DOI: 10.1002/asi

TABLE 3. Feature definitions for expansion terms.

| Feature template | ID | Type | Definition |
|---|---|---|---|
| Overall term frequency: $tf(T, C)$ | 1 | Integer | $tf(t_i, C)$: raw frequency of $t_i$ in $C$ |
| | 2 | Real | $log(tf(t_i, C))$ |
| Document frequency: $df(T, C)$ | 3 | Integer | $df(t_i, C)$: No. of documents in $C$ containing $t_i$ |
| | 4 | Real | $log(df(t_i, C))$ |
| Inverse document frequency: $idf(T, C)$ | 5 | Real | $log((N - df(t_i, C) + 0.5)/(df(t_i, C) + 0.5))$ |
| | 6 | Real | $log(idf(t_i, C))$ |
| Co-occurrence with query terms: $cooccurence(T, Q, C)$ | 7 | Integer | $\sum_{j=1}^{n} cooccurence(t_i, q_j, C)$ : co-occurrence of $t_i$ and $q_i$ |
| | 8 | Real | $log\left(\sum_{j=1}^{n} cooccurence(t_i, q_j, C)/n\right)$ |
| | 9 | Integer | $\sum_{k=1}^{l} cooccurence(t_i, pair(Q_k), C)$ : co-occurrence of $t_i$ and $pair(Q_k)$ |
| | 10 | Real | $log\left(\sum_{k=1}^{l} cooccurence(t_i, pair(Q_k), C)/l\right)$ |
| Term proximity: $proximity(T, Q, C)$ | 11 | Integer | $proximity(T, Q, C)$: No. of documents in $C$ containing $t_i$ and $q_i$ within a certain window size |
| Feedback term frequency: $tf(T, S)$ | 12 | Integer | $tf(t_i, S)$: raw frequency of $t_i$ in $S$ |
| | 13 | Real | $log(tf(t_i, S))$ |
| Term-dependence score | 14 | Real | The scores of terms based on term-dependence method |
| Co-occurrence with proximity: $cooccurence(T, Q, S)$ | 15 | Integer | $\sum_{j=1}^{n} cooccurence(t_i, q_j, S)$ : co-occurrence of $t_i$ and $q_i$ within a certain window size |
| | 16 | Integer | $\sum_{k=1}^{l} cooccurence(t_i, pair(Q_k), S)$ |
| *tf-idf* | 17 | Real | $tf(t_i, S) * idf(t_i, C)$ |
| | 18 | Real | $log(tf(t_i, S) * idf(t_i, C))$ |
| | 19 | Real | $log(tf(t_i, S) * df(t_i, C))$ |

frequency (*tf*) and document frequency (*df*) are related to the term itself and take into account the term occurrences and number of documents with a term. We define features 1–4 based on those to model term ubiquity and specificity, respectively. Inverted document frequency (*idf*) is widely used in vector space models to express the importance of a term. We define features 5 and 6 based on *idf*.

To capture the relationship between a term and a query, we define some features based on co-occurrence. Co-occurrence refers to two terms appearing in the same document. In other words, if a candidate term and a query term co-occur more times, the relationship between them would be closer. Furthermore, when a term co-occurs with a pair of query terms more times, it indicates that the term is more likely related to the query. This co-occurrence relation takes into consideration more query context and may be better evidence to present co-occurrence. We accumulate the instances of term co-occurrences with all the query terms and all the query term pairs to express the relationship between a term and a query. We define features 7–10 based on this idea.

Our experiments use the collections in standard TREC format, which contains several fields for each document. For example, each document in the Robust2004 collection includes the title field, the text field, and the dateline field. Terms appear in different fields and so gain distinct importance. For example, a term that appears in the title field obviously gains more importance in comparison with one appearing in the text field, and also the co-occurrence of a term and a query in the title of a document indicates that they are more relevant, so we extract features both from every single field and from the entirety of all the fields. That is to say, we draw features 1–10 from the title field, the text field, the dateline field, and all the fields as different features in the feature set.

Term proximity assumes two terms that co-occur more closely are more relevant to each other, which can help measure the importance of a term with respect to a query. So we define feature 11 as the number of documents in which two terms co-occur within a certain window size. The window size is empirically set to be 5 and 10 words, respectively, which is the same setting used by Lease et al. (2009).

In PRF, we obtain some feedback documents based on the query language model (Ponte & Croft, 1998) and assume that these documents are relevant to the query. Term information obtained from feedback documents may be useful for term ranking, so we define some features based on feedback documents. Features 12 and 13 are related to term frequencies in the feedback documents set. Feature 14 is the score of a term based on the TD method. Moreover, co-occurrence is considered in feedback documents in the definitions of features 15 and 16. Because the candidate term and query term co-occur in these documents, we count their co-occurrences when they appear within a certain window size, where the window size refers to the number of terms between them. The text window size used here is 15 words. Term frequency-inverse document frequency (*tf-idf*) is a numerical statistic that reflects how important a term is to a document in a collection. It is often used as a weighting factor in information retrieval and text mining. We define features 17–19 based on *tf-idf*.

In addition, because a given statistic will be more reliable after normalization, we normalize all the feature values to the interval [0–1]. As we extract features 1–10 from three


DOI: 10.1002/asi

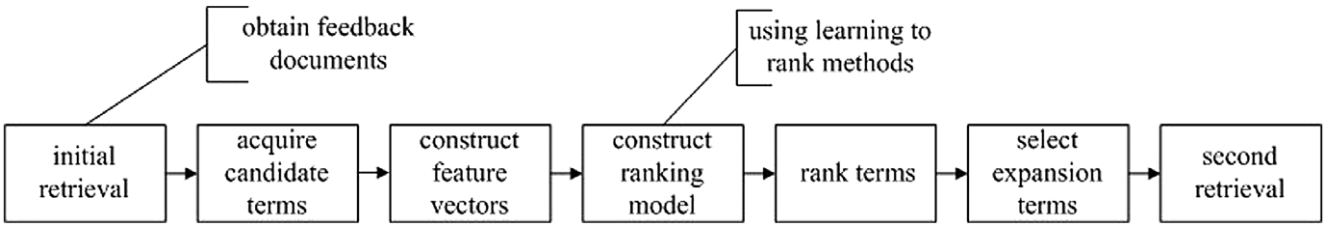


FIG. 1. Query expansion framework based on learning to rank.

single fields and the whole document, and features 11–19 from the whole document set, we finally obtain $(10 \times 4) + 9 = 49$ term features to represent a term as a feature vector.

*Learning to Rank Expansion Terms*

The learning to rank approach uses machine learning techniques to construct a ranking model automatically. The resulting model ranks new objects according to the relevance degrees assigned in the model. Specifically, the model for term ranking has two general properties. The first is that it is feature-based, which means that all the terms are represented as feature vectors that can reflect the relevance of the terms to the query. The other is that the learning process is a discriminative training process like traditional machine learning methods (Liu, 2009). A discriminative training process can combine different kinds of term features without the necessity for defining a probability about the terms and the correctness of prediction.

In particular, learning to rank is grouped into three approaches: the pointwise approach, the pairwise approach, and the listwise approach. Different approaches model the learning to rank process in different ways.

The pointwise approach is a straightforward way to solve ranking problems with machine learning technologies. When ranking with pointwise methods, one assumes that the exact relevance degree of each term is what we are going to predict, even though it may not be necessary when the target is to produce a ranked list of terms. Regression is used as a pointwise learning to rank method to reduce the ranking problem to a regression problem, which takes every single term feature vector as input and outputs the relevance degree of each term. We use the version of regression implemented in $SVM^{light}$ (Joachims, 1999) to investigate the effectiveness of the pointwise approach for query expansion.

The pairwise approach does not focus on accurately predicting the relevance degree of each term; instead, it concerns the relative order of two terms. In this sense, it is closer to the concept of ranking than the pointwise approach. RankSVM (Cao et al., 2006; Joachims, 2002) is a pairwise learning to rank algorithm, which utilizes support vector machine (SVM) for the ranking task. It inherits good properties from SVM because it is well rooted in the SVM framework, which formalizes learning to rank as a problem that classifies instances into categories with different relevance degrees. In our experiment, we use the implementation of RankSVM, $SVM^{rank}$, presented by Joachims (2006). RankBoost (Freund et al., 2003) combines preferences based on the boosting approach to machine learning. It utilizes the object pairs with preferences as instances in its training process. Similar to other boosting algorithms, RankBoost operates in rounds and works by combining many weak learners, which are weakly correlated with the target ranking model. The final ranking model of RankBoost is an ensemble of all the weak learners. We implement RankBoost according to Freund et al. (2003) and investigate the effectiveness of pairwise methods for term ranking based on RankSVM and RankBoost.

The listwise approach takes the ranking list as objects for calculating the difference between permutations using a listwise loss function or directly optimizing IR evaluation measures. We use LambdaMART (Burges, 2010; Wu et al., 2008) to investigate the listwise approach for term ranking. LambdaMART is the boosted tree version of LambdaRank (Burges et al., 2006), which is based on gradient boosting (Friedman, 2001), and has been shown to be among the best performing learning methods on public data sets in Track 1 of the 2010 Yahoo! Learning to Rank Challenge. LambdaMART combines MART and LambdaRank. The resulting model of MART is the linear combination of the outputs with respect to a set of regression trees. LambdaMART uses $\lambda$ as the gradient of loss function and uses a boosted regression tree as its model to decrease ranking loss in iterations as MART does. In our experiments, we use the version of LambdaMART implemented by Ganjisaffar, Caruana, and Lopes (2011). Readers can refer to Burges (2010) for the details of this algorithm.

We present our query expansion framework in Figure 1, where LTR methods refer to learning to rank methods. First, we conducted an initial retrieval and obtained a set of candidate expansion terms from feedback documents. Second, we extracted features and ground truth labels with respect to the terms to construct term feature vectors. Next, we trained term ranking models to refine the candidate terms. Finally, the resulting model was employed to rank terms and reformulate the original query.


DOI: 10.1002/asi

## Experiments and Analysis

### *Experimental Settings*

This section evaluates our query expansion framework on three standard TREC collections: Robust2004 (the data set of the TREC Robust Track started in 2003), WSJ87–90 (*Wall St. Journal*), and AP88–90 (Associated Press). We choose the three TREC collections to facilitate comparison with other work because these collections are public data sets. Since our framework is general, other collections can also be applied. We used all the topics in these collections for our experiments, in which title fields of the topics are used as the original queries. The average length of original queries for the Robust2004 collection is 2.55 words, whereas the average length for AP and WSJ collections is 5.15 words. Table 4 shows the statistics for these collections.

The Indri search engine (Strohman, Metzler, Turtle, & Croft, 2004) is used as the basic retrieval system in our experiments. From initial retrieval based on Indri-implemented language modeling, we take the top-$N$ documents in the result list as feedback documents and obtain $M$ candidate terms from them based on TD methods. We stem the terms with the Porter stemmer and remove the stop words using a standard *InQuery* stoplist in advance. Then, we rerank the terms using a term ranking model to form a new term ranking list, the top-$k$ terms of which will be selected for query expansion. The query is reformulated with the following query language structure of Indri:

$$\#weight(\lambda\ Q_{original}\ (1.0-\lambda)\ \#combine(weight_1\ term_1\ weight_2\ term_2\ \ldots\ weight_k\ term_k)), \quad (1)$$

where the first item is the weighted original query in the expanded query, and the second item is a list of expansion terms in which terms are weighted according to the scores given by the term ranking model. Term weights are normalized to the interval [0–1] to make fair comparisons for different ranking models because the term ranking scores obtained from different ranking models are quite varied. $\lambda$ is the weight for the original query. We fix the parameter as $M = 150,\ N = 10,\ k = 50,\ \lambda = 0.5$ because relatively good performance is achieved in this setting.

To evaluate the retrieval performance, we adopt MAP for the top 1,000 documents retrieved, precision on the top $k$ documents (P@k), and normalized discounted cumulative gain of the top $N$ documents (NDCG@k) as evaluation measures. In the term ranking experiments, for each collection, we divide its topics by query numbers into three parts, the training set, the validation set, and the testing set. We conduct fivefold cross-validation experiments, each using 60% of the queries for training a term ranking model, 20% for estimating the parameters, and 20% for predicting new queries. Four learning to rank methods, regression, RankBoost, RankSVM, and LambdaMART, are investigated for term ranking.

TABLE 4. Statistics of evaluation collections.

| Collection | No. of documents | Topics |
|---|---|---|
| Robust2004 | 528,155 | 301–450 |
| AP | 242,198 | 51–200 |
| WSJ | 173,252 | 51–200 |

### *Baseline Models*

This section introduces some baseline models used in our experiments. The first is the query-likelihood language model (QL; Zhai & Lafferty, 2001). The second is Lavrenko's relevance model (RM; Lavrenko & Croft, 2001), which selects the top $k$ most likely terms obtained from $N$ feedback documents to form an expanded query. We use the versions of these two models implemented in Indri. Furthermore, we compare our framework with the method presented by Cao et al. (2008), which treats a term selection problem as a term classification problem and can be seen as a strong baseline. Compared with term classification, query expansion based on term ranking adopts learning to rank approaches to obtain a sorted term list scoring from high to low. Our framework is developed to choose the most relevant expansion terms from the top of the list and give weight to each term based on the ranking score. We fix the experimental setting the same to facilitate the comparison between the term classification method and the term ranking method. In the classification method, first we choose a set of candidate expansion terms based on the TD method and then use the multiple relevance labeling strategy to label the terms. Next, we use the implementation of $SVM^{light}$ (Joachims, 1999) to train an SVM classifier for terms to choose the good terms for expansion. In the reformulated query, terms are weighted based on the posterior probability given by the classifier. We also conduct fivefold cross-validation for classification methods to facilitate the comparison. In the next few subsections, we give some details on how our query expansion framework works and some analysis of experimental results.

### *Performance of the TD Method*

Before evaluating the term ranking framework for term selection, let us first evaluate the performance of the methods based on FI and SD, respectively. We adopt linear interpolation to combine FI and SD as the TD method. We examine these three methods to choose a set of candidate terms for expansion and show the experimental results in Figure 2 for all topics in three TREC collections together with retrieval performance of baseline models in terms of MAP.

As we can see from Figure 2, the relevance model can significantly improve the retrieval performance over the query likelihood model. In comparisons of FI, SD, and TD methods, we can see that, although both the FI and SD methods boost retrieval performance of QL to a great extent, they perform as well as RM, and TD as the combination of


DOI: 10.1002/asi

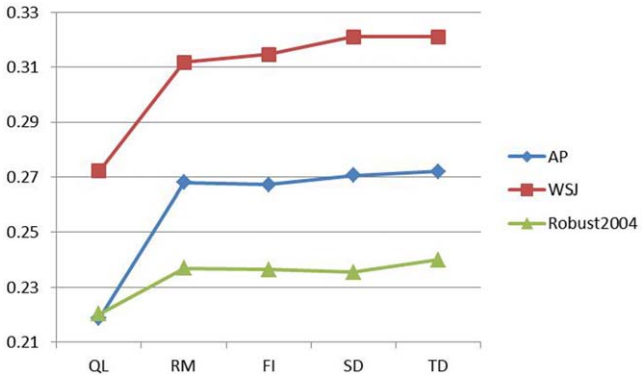


FIG. 2. Evaluation of the term-dependence method. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE 5. Evaluation of labeling strategies.

| Strategy | AP | WSJ | Robust2004 |
|---|---|---|---|
| Impact_*only* | 0.2792 | 0.3240 | 0.2292 |
| Impact_*k* | *0.2823* | *0.3307* | *0.2413* |

the two methods enhances the performance of RM further. The results show that the TD method can help choose a set of more relevant terms and boost retrieval performance applied for feedback documents and considering only that query terms are either fully independent or sequentially dependent with each other is not sufficient to express the relationship between terms and a query. The TD method combines the two comprehensively and thus has better performance. We fix the interpolation parameter for each collection as $\lambda_{WSJ} = 1.0$, $\lambda_{AP} = 0.7$, and $\lambda_{Robust} = 0.6$ because better retrieval performance can be achieved in this setting, which contributes more to further selection of terms.

### *Term Labeling Strategy Evaluation*

We have proposed two strategies to give relevance labels to candidate terms. The first labels terms merely according to their impact on retrieval performance, where a relevant term is the one that can improve retrieval performance when added to the query. We denote this strategy as Impact_only. The second labels terms in consideration of both the impact and the number of expansion terms parameterized as $k$, by which we label terms with three relevance degrees, definitely relevant, possibly relevant, and irrelevant. We denote this strategy as Impact_k.

This section compares and analyzes the experimental results with these two strategies. Table 5 shows the retrieval performance of term ranking models based on both strategies in terms of MAP on each collection, and LambdaMART is used for term ranking. To improve readability, we show the best results in italics for each method in Table 5, which is the same in the next subsections.

TABLE 6. Categories of features.

| Category | Feature ID | Category | Feature ID |
|---|---|---|---|
| TF_IDF | 1,2,5,6 | CoocTwo | 9,10 |
| DF | 3,4 | Feedback | 12–14,17–19 |
| CoocOne | 7,8 | Proximity | 11,15,16 |

From Table 5, we can see that Impact_k performs much better and more steadily than Impact_only. Our explanation is that, since number of candidate terms with positive impact varies widely on queries, top-ranked terms may not be effectively distinguished from others in the training process based on the Impact_only strategy, especially for the queries with more relevant candidate terms. The Impact_k strategy solves this problem by balancing the impact and the number of expansion terms, so it is more beneficial for learning to rank terms.

### *Feature Selection*

We have defined some term features in the previous section. They differ from each other in their effectiveness for term ranking, so we investigate their usefulness for term ranking in this section based on retrieval performance for query expansion. We examine the usefulness on the WSJ collection, and LambdaMART is used for training term ranking models.

First, we classify the features into six categories presented in Table 6 corresponding to the feature IDs in Table 3. We extend the features by extracting each of them from different fields as mentioned previously. In addition, with only one category of features for term selection, the learning method will not perform very well because the number of features is so low. Beyond that, we take TF_IDF features as the baseline feature set because *tf* and *idf* have been proved effective in IR for characteristic terms. We examine the usefulness of other feature sets by combining each of them with TF_IDF features. We report the experimental results in Table 7 in terms of MAP, P@k, and NDCG@k, where we denote NDCG@k as N@k.

From Table 7, we can see that, in comparison, three feature sets, Proximity, Feedback, and CoocTwo, have the top-3 best performance with the TF_IDF as a baseline feature set. We take these three feature sets together as the optimal feature set, denoted as Opti_Only. Next, we examine the performance of the optimal feature set combined with TF_IDF, denoted as Opti_TF_IDF. In addition, we examine the performance of the term ranking model with all the features.

Table 8 shows the performance of the ranking models based on the three feature sets described above. Compared with the feature set with all features and the Opti_TF_IDF feature set, the Opti_Only feature set has the best


DOI: 10.1002/asi

TABLE 7. Evaluation on the usefulness of feature sets in terms of retrieval performance.

| Feature set | MAP | P@3 | P@5 | P@10 | N@3 | N@5 | N@10 |
|---|---|---|---|---|---|---|---|
| TF_IDF | 0.3016 | *0.5578* | 0.5373 | 0.4853 | 0.5584 | 0.5476 | 0.5168 |
| DF + TF_IDF | 0.3079 | 0.5511 | 0.5400 | 0.4980 | 0.5526 | 0.5482 | 0.5263 |
| CoocOne + TF_IDF | 0.3051 | 0.5422 | 0.5227 | 0.4913 | 0.5506 | 0.5380 | 0.5200 |
| CoocTwo + TF_IDF | 0.3112 | 0.5555 | 0.5373 | 0.4927 | 0.5589 | 0.5481 | 0.5216 |
| Feedback + TF_IDF | 0.3233 | 0.5555 | *0.5453* | 0.5107 | 0.5607 | 0.5557 | 0.5379 |
| Proximity + TF_IDF | *0.3306* | 0.5534 | 0.5400 | *0.5187* | *0.5682* | *0.5589* | *0.5476* |

TABLE 8. Evaluation for an optimal feature set.

| Feature set | MAP | P@3 | P@5 | P@10 | N@3 | N@5 | N@10 |
|---|---|---|---|---|---|---|---|
| All Features | 0.3293 | *0.5889* | 0.5667 | 0.5200 | *0.5846* | 0.5739 | 0.5485 |
| Opti_TF_IDF | 0.3298 | 0.5600 | 0.5653 | *0.5226* | 0.5636 | 0.5694 | 0.5474 |
| Opti_Only | *0.3307* | 0.5756 | *0.5680* | 0.5173 | 0.5828 | *0.5785* | *0.5495* |

performance with fewer features, which contributes more to the efficiency of training the term ranking model.

A possible explanation is that features in the Opti_Only set have stronger capability both in expressing term relevance with the original query and in discriminating them from each other, and other features may have less ability than these features, which makes them less important and negligible in the training process. Therefore, we chose only the Opti_Only feature set to train other term ranking models in the next experiments.

TABLE 9. Term ranking accuracies evaluated by MAP.

| Methods | AP | WSJ | Robust2004 |
|---|---|---|---|
| TD | 0.5806 | 0.5721 | 0.4998 |
| Classification | 0.6029 | 0.5913 | 0.5389 |
| Regression | 0.6127* | 0.5958* | 0.5368 |
| RankBoost | 0.6163* | 0.6006* | 0.5419* |
| RankSVM | *0.6185** | 0.6057* | 0.5476* |
| LambdaMART | 0.6143* | *0.6083** | *0.5535** |

*Improvement of term ranking over classification method is significant with 95% confidence level ($P < 0.05$).

### *Term Ranking Accuracy*

This section evaluates our term ranking models with diverse ranking methods, Regression, RankBoost, RankSVM, and LambdaMART, in terms of term ranking accuracy. Because our query expansion framework aims at a further selection of expansion terms from a set of candidate terms, we show the ranking accuracy to see whether our term ranking models have the ability to find more relevant terms for expansion. Let us now examine the term ranking models. Specifically, we rank terms based on the score by each method and compare different ranking lists to test the term ranking accuracy. We adopt MAP as the evaluation measures in the experiment because MAP is more focused on ranking relevant terms on the top of the ranking list, and top-ranked terms will get more weight in expanded query, where the weight embodies the degree of term relevance. Table 9 shows the experimental results. We compare the results using statistical tests (i.e., two-tailed paired Student's $t$ tests), where an asterisk indicates that improvement of term ranking over classification method is significant with a 95% confidence level ($P < 0.05$).

From Table 9, we can see that, compared with the TD method, classification for terms can help choose more relevant terms, and the improvement in term accuracy is significant. Learning to rank approaches can contribute more for selecting relevant terms compared with the classification method and show significant improvement over classification method in terms of term ranking accuracy. Experimental results show that learning to rank a set of candidate terms can indeed help choose more relevant terms for expansion. In the next subsection, we examine whether term ranking can boost retrieval performance.

To illustrate our expansion term selection method better, we take the query numbered 51 as an example. The content of the query is "Airbus Subsidies," and we list the top 10 expansion terms obtained from both the classification model and the LambdaMART ranking model in the AP collection, together with the relevance judgment of each term, as shown in Table 10.

From Table 10, we can see that, for query 51, the LambdaMART ranking model achieves a more accurate term ranking list, which can help choose more relevant expansion terms. Higher ranked terms can acquire more weights in the expanded query, so our framework may enhance the retrieval performance using the term ranking model.


DOI: 10.1002/asi

TABLE 10. A toy example of expansion terms.

| Top-10 terms using classification | Relevance judgment | Top-10 terms using LambdaMART | Relevance judgment |
|---|---|---|---|
| Airbus | Relevant | Airbus | Relevant |
| Nations | Irrelevant | Subsidies | Relevant |
| Subsidies | Relevant | Agreement | Relevant |
| Agreement | Relevant | Decision | Relevant |
| Decision | Relevant | Yeutter | Relevant |
| Increase | Relevant | Provisions | Relevant |
| Helpful | Irrelevant | Coming | Irrelevant |
| Consortium | Irrelevant | Manufacturers | Relevant |
| Coming | Irrelevant | Statement | Relevant |
| Yeutter | Relevant | Nations | Irrelevant |

TABLE 11. Retrieval performance of term ranking models evaluated by MAP.

| Methods | AP | WSJ | Robust2004 |
|---|---|---|---|
| QL | 0.2187 | 0.2721 | 0.2202 |
| RM | 0.2696 | 0.3117 | 0.2381 |
| TD | 0.2713 | 0.3210 | 0.2393 |
| Classification | 0.2655 | 0.3231 | 0.2394 |
| Regression | 0.2757*† | 0.3130 | 0.2233 |
| RankBoost | 0.2789*† | 0.3207 | 0.2208 |
| RankSVM | 0.2791*† | 0.3214 | 0.2374 |
| LambdaMART | *0.2823*†* | *0.3307*†* | *0.2413*†* |
| TD + Oracle | 0.3822 | 0.4279 | 0.3485 |

*The MAP improvement is statistically significant over the TD method. †The MAP improvement is statistically significant over the classification method.

### *Performance of Term Ranking Models Based on Learning to Rank Methods*

This section evaluates term ranking models compared with baseline models, QL, RM, TD, and the classification method. In the term ranking process, we adopt Impact_k as the term relevance labeling strategy and feature set Opti_Only for training term ranking models. Precision evaluations (MAP) of results are shown in Table 11 with respect to the three collections. We conduct fivefold cross-validation both on term ranking models and on term classifications models, and the results reported in Table 11 are those averaged over five trials for each collection. Furthermore, we compare the retrieval performance using statistical tests with 95% confidence level ($P < 0.05$). Specifically, we perform significant tests over the retrieval performance of the term classification method and TD methods to test whether the improvement is significant, where an asterisk and a dagger indicate that the MAP improvement is statistically significant over the TD method and classification method, respectively.

As we can see from Table 11, the TD method outperforms baseline models, QL and RM, on all the collections. As stated above, more relevant candidate expansion terms can be found by selecting terms using the TD method from

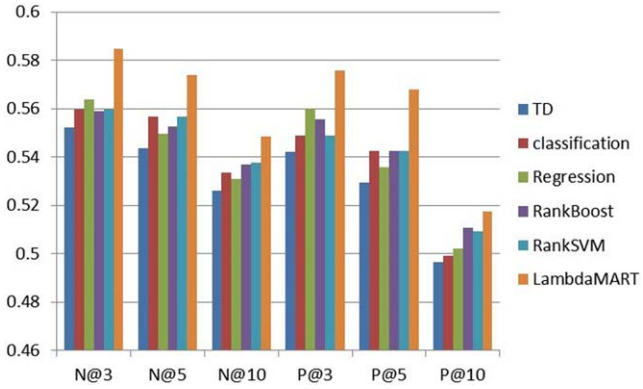


FIG. 3. Retrieval performance in terms of NDCG@k and P@k on the WSJ collection. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

feedback document. Furthermore, the classification method performs a little better than TD on WSJ and Robust2004 and performs a little worse on AP, so it is inconsistent compared with TD on the three collections. A similar tendency can be found in term ranking models based on regression and RankBoost in spite of some statistically significant improvement over the classification method on the AP collection. These three methods contribute less to retrieval performance in spite of improvements in term ranking accuracy. Term ranking model based on RankSVM performs better than the former two term ranking models, which has significant improvement on AP and significance ties on WSJ and Robust2004 compared with the classification method, so we find that term ranking based on RankSVM performs as well as the classification method and slightly better than the TD method. Term ranking based on LambdaMART performs the best overall and shows a consistently significant improvement over other methods on all the collections. Experimental results show that the term ranking model based on LambdaMART boosts term ranking accuracy compared with other methods shown in Table 9 and shows significant improvement on retrieval performance. Our explanation is that, because it chose more relevant terms and more reasonable term weights, higher term ranking accuracy can contribute more to the retrieval performance; that is, term ranking is effective in further selection of terms.

NDCG has proven effective in evaluating retrieval performance with a graded relevance scale, so we evaluate the results of different methods by NDCG. Figures 3–5 show the results evaluated by P@k and NDCG@k on each collection, where N@k denotes NDCG@k. On WSJ collection, compared with the TD method, term ranking models get better results. The classification method beats other methods except for the ranking model based on lambdaMART in terms of most evaluation measures. LamdbaMART performs the best. The same tendency can be seen on the AP collection, where TD performs the best as evaluated by P@3 and NDCG@3; that is, the top 3 documents returned by the TD method are more relevant on the collection. Lambda-


DOI: 10.1002/asi

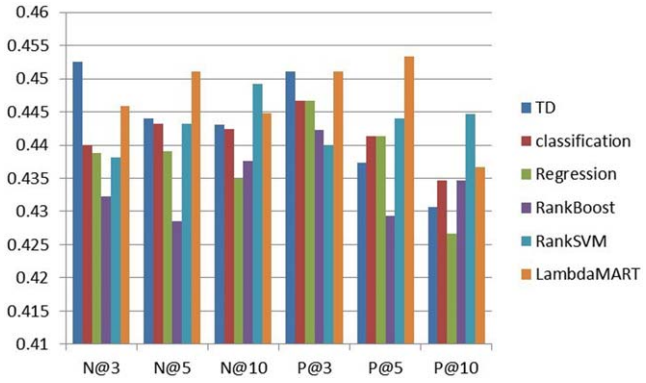


FIG. 4. Retrieval performance in terms of NDCG@k and P@k on the AP collection. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

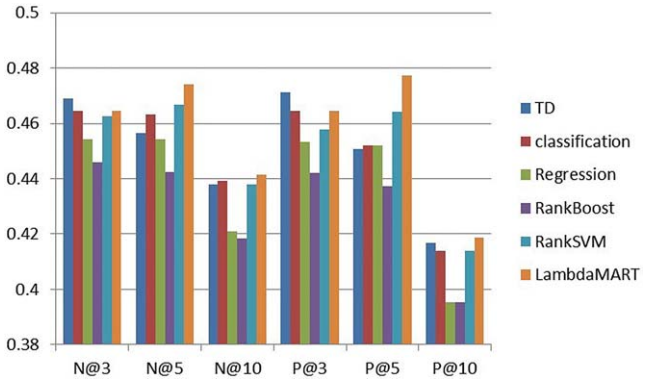


FIG. 5. Retrieval performance in terms of NDCG@k and P@k on the Robust2004 collection. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

MART and RankSVM have better performance by other measures. On the Robust2004 collection, we can get a clearer view of the best. Ranking models based on LamdbaMART as well as RankSVM performs the best compared with other methods.

Above all, we can see that different term ranking models based on learning to rank approaches have different effects on term selection. Pariwise term ranking models based on RankBoost and rankSVM are more effective than the pointwise method Regression and term ranking based on RankSVM perform better than RankBoost. The listwise term ranking model via LambdaMART achieves better results than pairwise methods. Retrieval performances for term ranking models based on LambdaMART and RankSVM are more consistent on all the collections. The analysis also shows that tree-based or non-inear models, such as ranking models based on LambdaMART and RankSVM, are more useful in term ranking for query expansion than the TD method or classification method. Expanded queries using the appropriate term ranking method can indeed choose more relevant expansion terms and boost retrieval performance.

TABLE 12. Values of parameters used for tuning.

| Methods | Parameter | Values |
|---|---|---|
| LambdaMART | *Max leaves* | 10, 15, 20, 25 |
| | *Min obs. per leaf* | 0.12, 0.25, 0.5 |
| | *Learning rate* | 0.025, 0.05, 0.1 |
| | *Training sampling* | 0.25, 0.5, 0.75, 1 |
| | *Feature sampling* | 0.25, 0.5, 0.75, 1 |
| RankSVM | *C* | 10, 20 . . . 100 |
| RankBoost | *Rounds* | 100, 200 . . . 1,000 |

### *Parameter Selection for Ranking Methods*

We use the implementation of LambdaMART with randomness presented by Ganjisaffar et al. (2011), in which a basic learner should be fit on a subsample of the training set drawn at random without replacement, and feature sampling is also introduced on each tree split. Specifically, we tune five parameters for the LambdaMART algorithm, including maximum number of leaves per tree, minimum observations allowed in each leaf, learning rate, training sampling rate, and feature sampling rate. For RankSVM we tune the parameter *C*, which indicates the trading-off margin size against training error. For RankBoost, we tune the number of weak learners for each fold, which is the round of iteration. Table 12 shows the values of parameters used for finding the optimal configuration of each algorithm. For LambdaMART, we use grid search to examine the performance of different combinations of parameters; for RankSVM, we adjust the parameter *C* ranging from 10 to 100, and for RankBoost we choose the number of weak learner from 100 to 1,000.

## Discussion

Based on the experimental results, we investigate our query expansion framework in this section, and include some statistical analysis, the machine learning techniques, time cost for model training, and parameter tuning processes that could help enhancement in our study.

First, we perform some statistical analysis on our experimental results. In the IR field, type I error relates to the evaluation measure recall, which counts the ratio of retrieved relevant documents, whereas type II error relates to the evaluation measure precision, which focuses more on the top-ranked documents. When a user uses a search engine, he or she usually focuses more on the top-ranked item and less on whether all the relevant documents have been retrieved. Therefore, we use mainly a precision measure, especially the MAP, to evaluate the retrieval performance in our experiments. Comparing all the methods in terms of MAP, we find that LambdaMART is more effective for term selection in our framework.

Our query expansion framework employs learning to rank methods in expansion term selection, which uses machine learning techniques to solve the ranking problem.


DOI: 10.1002/asi

TABLE 13. Term ranking model training time for each method on WSJ collection.

| Methods | Classification | Regression | RankBoost | RankSVM | LambdaMART |
|---|---|---|---|---|---|
| Time cost | 30.07s | 16.25s | 52.23s | 32.40s | 28.33s |

Based on our experiments, we find that methods such as classification and regression directly introduce machine learning techniques into term selection, achieving a performance comparable to that of the TD method, which can be taken as a modification of traditional PRF. Furthermore, pairwise methods, such as RankBoost and RankSVM, respectively introduce boosting and support vector machine techniques in consideration of pairwise term preference orders, achieving better retrieval performance. The listwise approach, LambdaMART, uses gradient boosting tree as machine learning technique, outperforming other methods. Therefore, we believe that the boosting approach, especially the tree-based boosting model, is preferable for term selection in our query expansion framework.

In addition to some traditional evaluation measures, we compare these methods in terms of computational time. We find that our framework can achieve comparable time cost of searching compared with the original PRF method except for some additional time for model training, but, for a searching system, term ranking models can be built in advance. Once term ranking models have been learned, our framework can achieve comparable time for a user to retrieve the information needed. We compare the model training time of different methods in Table 13, where the cost time is averaged over all the five folds on the WSJ collection.

From Table 13, we can see that the regression-based term ranking model takes the least time to train the ranking model, and classification-based term selection, together with term ranking models based on RankSVM and LambdaMART, takes comparable time. The RankBoost-based term ranking model takes the most time. Therefore, we believe that the LambdaMART-based term ranking model has the best performance in terms of both traditional evaluation measures and time cost for model training.

We tune the parameters $N$ and $k$ based on retrieval performance of the original PRF method, in which $N$ stands for the number of feedback documents and $k$ the number of expanded terms in an expansion query. The parameter $k$ is also used in the term labeling strategy, Impact_k. We tune the parameter $M$, the number of candidate terms, based on the term ranking accuracies of the candidate terms. Figures 6–8 show the parameter tuning. We can see that relatively good performance can be achieved when the parameters are set as $M = 150$, $N = 10$, and $k = 50$.

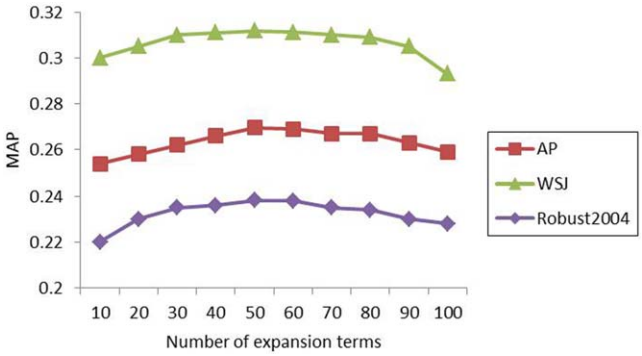


FIG. 6. Sensitivity of parameter $k$ (number of expansion terms) on the AP, WSJ, and Robust2004 collections. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

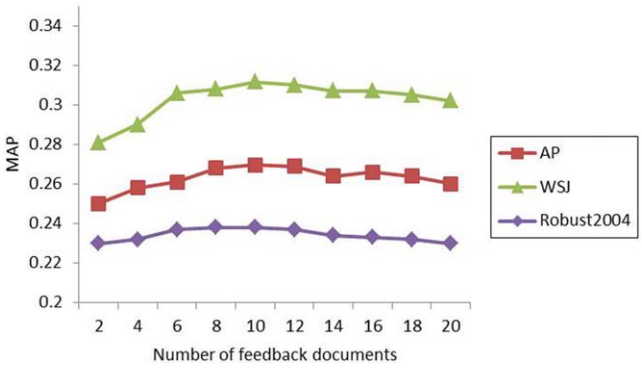


FIG. 7. Sensitivity of parameter $N$ (number of feedback documents) on the AP, WSJ, and Robust2004 collections. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

## Conclusions and Future Work

We have proposed a novel query expansion framework for document retrieval based on learning to rank. Specifically, learning to rank is introduced to rerank candidate terms obtained from feedback documents based on TD methods. Statistical features of expansion terms with respect to corresponding original queries are taken advantage of during training a term ranking model, and further expansion terms are selected based on the new term ranking list. We compare and analyze the effectiveness of term ranking models based on different learning to rank approaches. In addition, we compare the performance of two labeling strategies on terms and find that the strategy based on impact and number of expansion terms is better for term ranking. We examine the usefulness of various feature sets according to the effectiveness of term ranking models based on them and choose the optimal feature set for term ranking. Experiments on three TREC collections show that integrating learning to rank algorithms into query expansion can effectively


DOI: 10.1002/asi

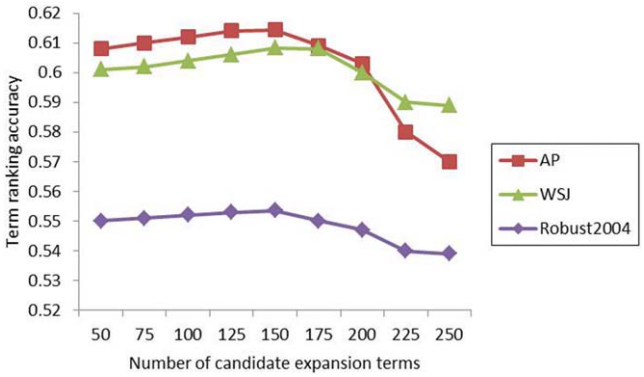


FIG. 8. Sensitivity of parameter *M* (number of candidate expansion terms) on the AP, WSJ, and Robust2004 collections. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

improve retrieval performance. The term ranking model based on LambdaMART performs the best for expansion term selection. Our future work will explore several directions, finding more useful features for expansion term selection in the term ranking process and investigating more efficient and effective learning to rank algorithms for term ranking of query expansion.

## Acknowledgments

This work is partially supported by grant from the Natural Science Foundation of China (No. 61277370, 61402075), Natural Science Foundation of Liaoning Province, China (No. 201202031, 2014020003), State Education Ministry and The Research Fund for the Doctoral Program of Higher Education (No. 20090041110002), the Fundamental Research Funds for the Central Universities.

## References

Buckley, C. (1994). Automatic query expansion using smart: Trec 3. In D.K. Harman (Ed.), In Proceedings of the third Text REtrieval Conference (TREC-3) (pp. 69–80). Gaithersburg, MD: National Institute of Standards and Technology.

Burges, C.J. (2010). From ranknet to lambdarank to lambdamart: An overview. Microsoft Research Technical Report MSR-TR-2010-82.

Burges, C.J., Ragno, R., & Le, Q.V. (2006). Learning to rank with nonsmooth cost functions. In B. Schölkopf, J.C. Platt, & T. Hoffman (Eds.), Proceedings of the 20th Annual Conference on Neural Information Processing Systems. NIPS (pp. 193–200). Cambridge, MA: MIT Press.

Cao, G., Nie, J.Y., Gao, J., & Robertson, S. (2008). Selecting good expansion terms for pseudo-relevance feedback. In S.H. Myaeng, D.W. Oard, F. Sebastiani, T.S. Chua, & M.K. Leong (Eds.), Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 243–250). New York, NY: ACM.

Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., & Hon, H.W. (2006). Adapting ranking svm to document retrieval. In E.N. Efthimiadis, S. Dumais, D. Hawking, & K. Järvelin (Eds.), Proceedings of the 29st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 186–193). New York, NY: ACM.

Cronen-Townsend, S., Zhou, Y., & Croft, W.B. (2004). A framework for selective query expansion. In D.A. Grossman, L. Gravano, C.X. Zhai, O. Herzog, & D.A. Evans (Eds.), Proceedings of the 13st ACM International Conference on Information and Knowledge Management (pp. 236–237). New York, NY: ACM.

Freund, Y., Lyer, R.D., Schapire, R.E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, 4(1), 933–969.

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.

Ganjisaffar, Y., Caruana, R., & Lopes, C.V. (2011). Bagging gradient-boosted trees for high precision, low variance ranking models. In W.Y. Ma, J.Y. Nie, R.A. Baeza-Yates, T.S. Chua, & W.B. Croft (Eds.), Proceedings of the 34st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 85–94). New York, NY: ACM.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In B. Schölkopf, C.J.C. Burges, & A.J. Smola (Eds.), Advances in kernel methods (pp. 169–184). Cambridge, MA: MIT Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In O.R. Zaïane & R. Goebel (Eds.), Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD (pp. 133–142). New York, NY: ACM.

Joachims, T. (2006). Training linear svms in linear time. In L. Ungar & T. Eliassi-Rad (Eds.), Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD (pp. 217–226). New York, NY: ACM.

Jones, R., & Fain, D.C. (2003). Query word deletion prediction. In C. Clarke & G. Cormack (Eds.), Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 435–436). New York, NY: ACM.

Kumaran, G., & Allan, J. (2008a). Effective and efficient user interaction for long queries. In S.H. Myaeng, D.W. Oard, F. Sebastiani, T.S. Chua, & M.K. Leong (Eds.), Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 11–18). New York, NY: ACM.

Kumaran, G., & Allan, J. (2008b). Adapting information retrieval systems to user queries. Information Processing & Management, 44(6), 1838–1862.

Lavrenko, V., & Croft, W.B. (2001). Relevance based language models. In W.B. Croft, D.J. Harper, D.H. Kraft, & J. Zobel (Eds.), Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 120–127). New York, NY: ACM.

Lease, M. (2009). An improved markov random field model for supporting verbose queries. In J. Allan, J.A. Aslam, M. Sanderson, C.X. Zhai, & J. Zobel (Eds.), Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 476–483). New York, NY: ACM.

Lease, M., Allan, J., & Croft, W.B. (2009). Regression rank: Learning to meet the opportunity of descriptive queries. In M. Boughanem, C. Berrut, J. Mothe, & C. Soulé-Dupuy (Eds.), Advances in Information Retrieval (pp. 90–101). Berlin, Heidelberg: Springer.

Lee, C.J., Chen, R.C., Kao, S.H., & Cheng, P.J. (2009). A term dependency-based approach for query terms ranking. In D.W. Cheung, I.Y. Song, W.W. Chu, X. Hu, & J.J. Lin (Eds.), Proceedings of the 18th ACM International Conference on Information and Knowledge Management (pp. 1267–1276). New York, NY: ACM.

Lee, K.S., Croft, W.B., & Allan, J. (2008). A cluster-based resampling method for pseudo-relevance feedback. In S.H. Myaeng, D.W. Oard, F. Sebastiani, T.S. Chua, & M.K. Leong (Eds.), Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 235–242). New York: ACM.

Lin, Y., Lin, H., Jin, S., & Ye, Z. (2011). Social annotation in query expansion: a machine learning approach. In W.Y. Ma, J.Y. Nie, R.A. Baeza-Yates, T.S. Chua, & W.B. Croft (Eds.), Proceedings of the 34stAnnual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 405–414). New York, NY: ACM.

Liu, T.Y. (2009). Learning to rank for information retrieval. Hanover, MA: Foundations and Trends in Information Retrieval.


DOI: 10.1002/asi

Lv, Y., Zhai, C., & Chen, W. (2011). A boosting approach to improving pseudo-relevance feedback. In W.Y. Ma, J.Y. Nie, R.A. Baeza-Yates, T.S. Chua, & W.B. Croft (Eds.), Proceedings of the 34st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 165–174). New York, NY: ACM.

Metzler, D., & Croft, W.B. (2007). Latent concept expansion using markov random fields. In W. Kraaij, A.P. de Vries, C.L.A. Clarke, N. Fuhr, & N. Kando (Eds.), Proceedings of the 30st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 311–318). New York, NY: ACM.

Oliveira, V., Gomes, G., Belém, F., Brandão, W., Almeida, J., Ziviani, N., & Gonçalves, M. (2012). Automatic query expansion based on tag recommendation. In X. Chen, G. Lebanon, H. Wang, & M.J. Zaki (Eds.), Proceedings of the 21st ACM International Conference on Information and Knowledge Management (pp. 1985–1989). New York, NY: ACM.

Ponte, J.M., & Croft, W.B. (1998). A language modeling approach to information retrieval. In W.B. Croft, A. Moffat, C.J. Rijsbergen, R. Wilkinson, & J. Zobel (Eds.), Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 275–281). New York, NY: ACM.

Qin, T., Liu, T.Y., Xu, J., & Li, H. (2010). Letor: A benchmark collection for research on learning to rank for information retrieval. Information Retrieval, 13(4), 346–374.

Robertson, S.E., Walker, S., Beaulieu, M., Gatford, M., & Payne, A. (1996). Okapi at trec-4. In D.K. Harman (Ed.), Proceeding of the Fourth Text Retrieval Conference. NIST Special Publication (pp. 73–97). Gaithersburg, MD: National Institute of Standards and Technology.

Rocchio, J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), The Smart Retrieval System (pp. 313–323). Englewood Cliffs, NJ: Prentice-Hall.

Strohman, T., Metzler, D., Turtle, H., & Croft, W.B. (2004). Indri: A language model-based search engine for complex queries. In K. Masback (Ed.), Proceedings of the International Conference on Intelligence Analysis (pp. 2–6). Washington, DC: Central Intelligence for Analysis and Production.

Sun, J., Wang, S., Gao, B.J., & Ma, J. (2012). Learning to rank for hybrid recommendation. In X. Chen, G. Lebanon, H. Wang, & M.J. Zaki (Eds.), Proceedings of the 21st ACM International Conference on Information and Knowledge Management (pp. 2239–2242). New York, NY: ACM.

Surdeanu, M., Ciaramita, M., & Zaragoza, H. (2008). Learning to rank answers on large online qa collections. In K. McKeown, J.D. Moore, S. Teufel, J. Allan, & S. Furui (Eds.), Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics. ACL (pp. 719–727). Cambridge, MA: MIT Press.

Tao, T., & Zhai, C. (2006). Regularized estimation of mixture models for robust pseudo-relevance feedback. In E.N. Efthimiadis, S. Dumais, D. Hawking, & K. Järvelin (Eds.), Proceedings of the 29st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 162–169). New York, NY: ACM.

Wu, Q., Burges, C.J., Svore, K.M., & Gao, J. (2008). Ranking, boosting, and model adaptation. Microsoft Research Technical Report MSR-TR-2008-109.

Xu, J., & Croft, W.B. (1996). Query expansion using local and global document analysis. In H.P. Frei, D. Harman, P. Schäuble, & R. Wilkinson (Eds.), Proceedings of the 19st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 4–11). New York, NY: ACM.

Xu, J., & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In W. Kraaij, A.P. de Vries, C.L.A. Clarke, N. Fuhr, & N. Kando (Eds.), Proceedings of the 30st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 391–398). New York, NY: ACM.

Zhai, C., & Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. In C. Pu, D. Grossman, & H. Paques (Eds.), Proceedings of the Tenth International Conference on Information and Knowledge Management (pp. 403–410). New York, NY: ACM.


DOI: 10.1002/asi