Ranking Specialization for Web Search: A Divide-and-Conquer Approach by Using Topical RankSVM

Jiang Bian*
College of Computing
Georgia Institute of Technology
jbian@cc.gatech.edu

Xin Li, Fan Li, Zhaohui Zheng
Yahoo! Labs
{xinli, fli, zhaohui}@yahoo-inc.com

Hongyuan Zha
College of Computing
Georgia Institute of Technology
zha@cc.gatech.edu

ABSTRACT

Many ranking algorithms applying machine learning techniques have been proposed in informational retrieval and Web search. However, most of existing approaches do not explicitly take into account the fact that queries vary significantly in terms of ranking and entail different treatments regarding the ranking models. In this paper, we apply a divide-and-conquer framework for ranking specialization, i.e. learning multiple ranking models by addressing query difference. We first generate query representation by aggregating ranking features through pseudo feedbacks, and employ unsupervised clustering methods to identify a set of ranking-sensitive query topics based on training queries. To learn multiple ranking models for respective ranking-sensitive query topics, we define a global loss function by combining the ranking risks of all query topics, and we propose a unified SVM-based learning process to minimize the global loss. Moreover, we employ an ensemble approach to generate the ranking result for each test query by applying a set of ranking models of the most appropriate query topics. We conduct experiments using a benchmark dataset for learning ranking functions as well as a dataset from a commercial search engine. Experimental results show that our proposed approach can significantly improve the ranking performance over existing single-model approaches as well as straightforward local ranking approaches, and the automatically identified ranking-sensitive topics are more useful for enhancing ranking performance than pre-defined query categorization.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Retrieval models*; H.4.m [**Information Systems**]: Miscellaneous—*Machine learning*

General Terms

Algorithms, Experimentation, Theory

Keywords

Ranking specialization for Web search, ranking-sensitive query topic, topical RankSVM

*The work was done when the first author was intern at Yahoo! Labs

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

1. INTRODUCTION

Ranking has become a central research problem for informational retrieval and Web search, since it directly influences the relevance of the search results, the quality of a search system and users' search experience. The task of ranking in the search process can be briefly described as follows. Given a query, the deployed ranking function measures the relevance of each document to the query, sorts all documents based on their relevance scores, and presents a list of top-ranked ones to the user. Thus, the essential problem of search technology is to design a ranking function that can best represent relevance. In the past, many models have been proposed, including the Boolean model [2], vector space model [18], probabilistic model [17] and language modeling method [14]. Most recently, there are renewed interests in exploring the techniques from machine learning for building ranking functions, some popular algorithms including MCRank [15], RankNet [6], RankSVM [12], RankBoost [8], GBRank [23], ListNet [7],and IsoRank [24].

In most of the previous work, the significant difference in queries is not adequately addressed in the context of ranking. This is clearly not appropriate, particularly for Web search, since queries vary largely in multiple facets, such as semantics, users' search intention, popularity, length, number of relevant documents, etc. For example, queries can be related to various semantical domains, such as product, travel, and autos, or be categorized as navigational, informational, and transactional [5] in terms of search intentions; queries can vary in terms of their popularity, i.e. the frequency of occurrence in the search log; queries can also be different in length; and at some time, queries are classified as those associated with many relevant documents and those having very few relevant documents.

These various types of query difference make it difficult to build a single general ranking function for all kinds of queries, because the ranking function, while indicating good ranking relevance for a certain type of queries, may not be able to achieve similar performance for other types of queries. For instance, many current search engines can achieve good ranking performance on general short queries with 2-3 query terms, but they usually can not achieve high relevance for long queries with more query terms. And, for popular queries that are often searched by Web users, search engines can generally return good results, but for those that rarely occur or are new phrases on the Web, search engines may not be able to give sound ranking relevance. Before exploring new ranking methods to resolve these difficulties, we need to first investigate why query difference can result in such ranking problems.

We hypothesize the reason as the diverse feature impacts on ranking relevance with respect to different queries. For instance, for homepage finding (a kind of navigational) queries, the textual similarity between the query and the document title may be the best indicator of ranking relevance, whereas for topic distillation [21] (a kind of informational) queries, the whole-document TFIDF and BM25 features may be better for inferring relevance. As another example, for popular queries, document popularity features, such as PageRank, may be important for ranking, whereas for rare queries, it is not necessary to use document popularity to measure the ranking relevance. As a conclusion, a single ranking model cannot reflect different feature impacts for different queries, such that it would not be adequate to use one single ranking model for diverse types of queries.

Therefore, it is necessary to find some new approaches which can give better ranking relevance for all the different types of queries. A straightforward method is to add query difference in terms of additional features into learning the single ranking function, however, since this method requires high quality of both the new features and training data, it usually does not effective in practice (as shown in experiment in Section 5.1.2). In this paper, we propose a divide-and-conquer framework for improving the ranking relevance for all queries. The basic idea is to first identify a set of ranking-sensitive query topics and *divide* the problem of learning one single ranking model for all queries into learning a set of sub-models for corresponding different query topics. Then, we *conquer* these learning problems by introducing a global loss function and exploring a unified approach to co-optimize all sub-models. At testing time, we select a set of ranking-sensitive query topics the new query most likely belongs to, and apply respective ranking models to ranking the documents, then we assemble these ranking results together to obtain the final ranking for the new query. We name the whole framework as ranking specialization.

By applying ranking specialization, we can benefit ranking relevance in several ways: Firstly, ranking specialization can help to improve the overall search relevance, since the divide-and-conquer approach allows us to create and use topic-specific features and training data for learning the dedicated ranking model for each ranking-sensitive query topic. Secondly, we observed that, the current learning-to-rank algorithms, exploited to train a single ranking model, usually improve relevance on some queries, but hurt relevance on other queries compared with some baseline approach. Ranking specialization allows us to take deep-dive analysis and improve relevance on a subset of queries, without hurting others, which is the key for continuously improving Web search relevance. Furthermore, ranking specialization allows us to incrementally update the ranking models for search engines. From deep dive analysis, after we identify a group of queries that share the same ranking problems, we can easily update the model for the existing ranking-sensitive query topics, or add the queries as a new query topic, and then build a dedicated model for them.

However, there are three major challenges for our framework. The first one is how to identify the ranking-sensitive query topics. Most of previous query classification focus on categorizing queries in terms of semantics [3, 19], but such query classification may not be best for the purpose of improving ranking. Some of query taxonomy [5] is based on search intent [5], but they may not be fine-grained enough for ranking purpose. Furthermore, pre-defined query categorization may not even be available at learning time. In this paper, in addition to pre-defined query categories, we propose to identify ranking-sensitive query topics, in the sense that, each ranking-sensitive query topic represents a group of queries having the similar set of important features for measuring ranking relevance, and different query topics reflect diverse feature impacts on ranking. Due to the high correlation between the ranking-sensitive query topics and ranking features, we identify ranking-sensitive query topics by taking advantage of both ranking features of query's pseudo feedbacks and the prior knowledge of importance of ranking features.

The second major challenge is how to train the ranking model with respect to each ranking-sensitive query topics. Previous works, especially on local ranking or query-dependent ranking [13, 9], have proposed to train the ranking model for each query category separately. Although having achieved better ranking performance than single-model approaches, these methods do not consider dependency between different query categories, which may be beneficial to further improving ranking. Moreover, since they use only a small part of the training data for learning each model, it may cause the declining accuracy due to the lack of enough training examples. We instead propose a unified SVM-based method to learn all the models of all ranking-sensitive query topics simultaneously. In particular, we define a new global loss function by combining the ranking risks of all the training examples (for all query topics) with different weights according to the training query's similarity to different query topics. Intuitively, if one training query is highly correlated to a certain query topic, training examples with respect to this query will contribute more to learn the ranking model of this particular query topic. We name this learning method as *Topical RankSVM*.

The last challenge is how to conduct ranking for new testing queries. In our paper, we employ a testing method consistent with the training process. We first select a set of ranking models based on the correlation between the test query and the corresponding ranking-sensitive query topics. To obtain the final ranking result, we then aggregate the ranking scores computed by all selected models with weights based on the similarity between the test query and query topics. The intuition is that the ranking relevance of the test query depends more on the ranking models for the query topics that the test query most likely belongs to.

To evaluate the effectiveness of our proposed approaches for learning ranking functions, we conduct experiments on both Letor [16], a public benchmark dataset for learning ranking functions, and a large scale dataset from a commercial search engine. Experimental results show that our proposed approaches can significantly improve the performance of ranking over the single ranking model approach on both benchmark dataset and commercial search engine dataset. We also observe that our approach can out-perform previous local ranking or query-dependent ranking approaches. Furthermore, we provide analysis and conduct additional experiments to gain deeper understanding on the advantage of incorporating ranking-sensitive query categories.

The remaining parts of this paper are organized as follows. Section 2 introduces related work. Section 3 presents our divide-and-conquer ranking framework and its three major parts in details. Experimental setup and results are demonstrated in Section 4 and 5, respectively. We conclude the paper and point out future research directions in Section 6.

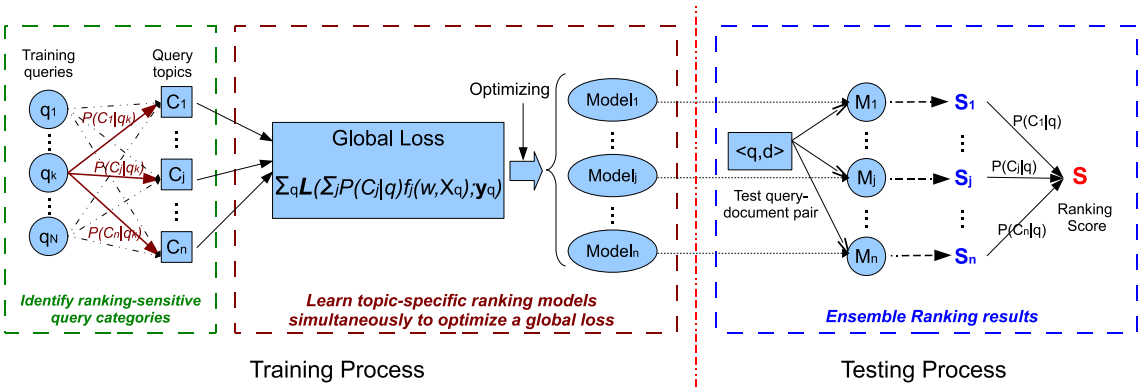


Figure 1: The ranking specialization framework for improving search relevance

2. RELATED WORK

Some of recent works have realized the importance and necessity of incorporating query difference into learning the ranking function. Zha et al. [22] propose an aTVT algorithm which implicitly incorporates query difference using monotonic transformations of the learned ranking functions. This approach focuses on the boundary of each query without considering broader query grouping. Kang et al. [13] classify queries into two categories (navigational and informational) and build two corresponding ranking models separately. However, it requires the availability and high accuracy of query classification. In a most recent work, Geng et al. [9] propose a K-Nearest Neighbor (KNN) method to employ different ranking models for different queries. Specifically, each training query holds a ranking model which is learned using the query itself and its neighboring queries. Given a test query, they find the most similar training query and use the corresponding model for ranking. Training time of this method is quite large, since many models need to be trained separately. And each model is trained using only a part of whole training set, which may cause the declining accuracy due to the lack of adequate training examples. In our work, we employ a divide-and-conquer approach for ranking specialization to improve the ranking relevance. We will discuss our approach in details in the following sections.

3. RANKING SPECIALIZATION

In this section, we will explore a new divide-and-conquer approach for ranking specialization for improving search relevance. We will start with introducing the general divide-and-conquer framework, followed by concrete discussion on three major parts of the framework in sequence.

3.1 A Divide-and-Conquer Framework

The divide-and-conquer framework consists of three major steps. In the first step, we target at identifying a set of ranking-sensitive query topics, $\mathcal{C}_{\text{query}} = \{C_1, C_2, \cdots, C_n\}$, based on all of training queries, $\mathcal{Q}_{\text{train}} = \{q_1, q_2, \cdots, q_N\}$. The recognized query topics are considered ranking-sensitive in the sense that different queries of the same topic should have similar characteristics in terms of ranking, especially, the similar family of important features for ranking. After this step, for each training query $q_i \in \mathcal{Q}_{\text{train}}$, we can obtain its distribution over all of extracted ranking-sensitive query topics, i.e. $\text{Topic}(q_i) = \langle P(C_1|q_i), P(C_2|q_i), \cdots, P(C_n|q_i)\rangle$, where $P(C_i|q)$ is the probability that q belongs to C_i. This step is like *dividing* the problem of learning one single ranking model for all training queries into a set of sub-problems of learning the ranking model for each ranking-sensitive query topic. The number of query topics can be set either empirically to constants, or through gap statistic [20].

The second step is to develop a unified learning method for learning multiple ranking models M_k $(k = 1, 2, \cdots, n)$, each exclusively corresponding to one ranking-sensitive query topic $C_k \in \mathcal{C}_{\text{query}}$. In our work, we propose a global loss function by combining risks of different ranking-sensitive query topics and introduce *Topical RankSVM* to train all the models $M_1, M_2, \cdots, M_n$, simultaneously, by minimizing the global loss function. By applying this unified learning method, we have considered dependency between different query topics when building their respective ranking functions, which can be beneficial to further improve ranking. Moreover, though treated unequally in learning each ranking function, all the training queries contribute to learn all ranking models of query topics, which avoids the lacking of training examples for learning the model of any single query topic. This unified method is quite general as we can use different feature set for different query topics. As incorporating information of query topics into the ranking algorithm, this step is like *conquering* the problem of learning the respective ranking model for each query topic.

The goal of the last step is to conduct ranking for new testing queries, $\mathcal{Q}_{\text{test}} = \{q_1, q_2, \cdots, q_t\}$. For each testing query $q_j \in \mathcal{Q}_{\text{test}}$, we apply an ensemble method, which try to minimize the risk consistent with the loss in training process. We first select a certain number H of ranking models M_{j1}, M_{j2},$\cdots$, M_{jH}, whose corresponding query topics hold H highest correlation with q_j, and then aggregating the ranking results S_{j1}, S_{j2},$\cdots$, S_{jH} obtained by M_{j1}, M_{j2},$\cdots$, M_{jH} into a final ranking results $\boldsymbol{S}$. After *divide-and-conquer*, this step aggregates ranking results from sub-models together into the improved final ranking results.

We summarize the general framework in Figure 1. Such ranking specialization approach allows us to use different feature sets or data sets to learn respective ranking models for different query topics, so as to boost the relevance for each query groups; And, the global loss in the second step serves as a unified relevance target when training different models for different query topics, such that we can optimize the overall search relevance when we train different ranking models. In the rest of this section, we will discuss the details of each step of the framework in sequence.

3.2 Identifying Ranking-Sensitive Query Topics

3.2.1 *Generating Query Features*

To identify ranking-sensitive query topics, we first generate a set of features to represent queries by taking ad-

vantage of the ranking features of top pseudo feedbacks of the query. For each training query $q \in \mathcal{Q}_{\text{train}}$, we retrieve a set of pseudo feedbacks, $PF(q) = \{d_1, d_2, \cdots, d_T\}$, consisting of the top T documents ranked by a reference model (we use BM25 in this paper). The ranking features of query-document pair of $\langle q, d_i \rangle$ are defined as a feature vector $\mathbf{x}^{qd_i} = \langle x_1^{qd_i}, x_2^{qd_i}, \cdots, x_D^{qd_i} \rangle$ (D is the number of ranking features). To represent q in a feature space, we aggregate the ranking features of top-T pseudo feedbacks of q into a new feature vector. We take the *mean* and *variance* of the ranking feature values as two aggregation methods. Thus, the feature vector of query q can be represented as

$$\langle \mu_1(q), \mu_2(q), \cdots, \mu_D(q), \sigma_1^2(q), \sigma_2^2(q), \cdots, \sigma_D^2(q) \rangle$$

where $\mu_k(q)$ denotes the mean value of k-th feature over q's pseudo feedbacks, and $\sigma_k^2(q)$ denotes the variance value of k-th feature over q's pseudo feedbacks.

In this paper, we will employ linear SVM model as the ranking algorithm. Thus, before generating query features, we have applied quantile normalization [4] on ranking features of query-document pairs in both training and testing dataset, such that the values of all ranking features are in the scale of $[0, 1]$. As a result, the values of extracted query features are also in the scale of $[0, 1]$.

3.2.2 Generating Query Topics and Computing Topic Distribution for Queries

After generating query features, we employ the mixture model as the clustering method to obtain ranking-sensitive query topics. We can either empirically set the number of cluster as a constant n, or determine it through gap statistic [20]. After learning the model, we can obtain a set of query clusters $\{C_1, C_2, \cdots, C_n\}$, where each cluster C_k can be represented as a vector $\mathbf{x}^{C_k} = \langle x_1^k, x_2^k, \cdots, x_{D_q}^k \rangle$, and D_q denotes the number of query features. We consider each cluster as one query topic.

Furthermore, we incorporate the prior knowledge of feature importance for ranking into identifying ranking-sensitive query topics. We obtain feature importance scores by using the ranking weights learned by a general RankSVM on a sample of training data. In our paper, we integrate the feature importance as weights into aggregated query features. Assuming the feature importance scores are $\mathbf{w} = \langle w_1, w_2, \cdots, w_{D_q} \rangle$, the weighted feature vector of query q_i is computed as

$$\mathbf{w} \odot \mathbf{x}^i \equiv \langle w_1 x_1^i, w_2 x_2^i, \cdots, w_{D_q} x_{D_q}^i \rangle \tag{1}$$

where $\mathbf{x}^i = \langle x_1^i, x_2^i, \cdots, x_{D_q}^i \rangle$ is the original feature vector of query q_i. Based on the new query features weighted by feature importance, we can employ the mixture model to obtain the ranking sensitive query topics with integrated feature importance.

Based on this representation of query topics, we are able to calculate the topic distribution $\text{Category}(q) = \{P(C_1|q), P(C_2|q), \cdots, P(C_n|q)\}$ for query q as

$$P(C_k|q) = \frac{|\mathbf{x}^q - \mathbf{x}^{C_k}|^2}{\sum_{i=1}^{n} |\mathbf{x}^q - \mathbf{x}^{C_i}|^2} \tag{2}$$

Since we take advantage of ranking features of top retrieved pseudo feedbacks of the query to generate query features as well as we compute topic distribution for queries by incorporating prior knowledge of feature importance for ranking, we consider the identified query topics and computed topic distribution for each query as ranking-sensitive.

3.3 A Unified Approach for Learning Multiple Ranking Models

3.3.1 Problem Statement

For traditional ranking approach, the task is to find a function f in the form of

$$y = f(X, \omega), \ f \in \mathcal{F}$$

where X denotes an $M \times D$ matrix representing D dimensional feature vectors of M documents; ω represents the unknown ranking parameters; y is a vector representing ranking scores of the M documents. The goal of learning is to select a best function $\hat{f}$, such that $\hat{f}$ minimizes the given loss function:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} L(f(X_i, \omega), y_i) \tag{3}$$

where N is the number of queries in the training set; X_i denotes the set of documents associated with the i-th query; y_i is the vector of corresponding ranking scores; L denotes a defined query-level loss function. Clearly, traditional ranking approach learns single ranking function for all queries.

Inspired by the diverse ranking characteristics implied by different queries, to improve ranking relevance, we formalize a new problem of learning multiple ranking functions, $f_1, f_2, \cdots, f_n \in \mathcal{F}$, given the identified ranking-sensitive query topics $C_1, C_2, \cdots, C_n$, where each ranking model $f_i (i = 1, \cdots, n)$ can represent the ranking characteristics of its corresponding query topic C_i. In order to create a unified relevance target for all topic-specific ranking models and let all training examples contribute to all ranking models, we propose a new global loss function by combining ranking risks of all training examples with different weights according to the training query's similarity to different query topics. By optimizing this global loss function, we can learn multiple ranking functions, simultaneously. The intuition is that if the query of one training example is highly correlated to a certain query topic, this training example will contribute more to learn the ranking function of this query topic. Specifically, the global function is defined as

$$\langle \hat{f}_1, \cdots, \hat{f}_n \rangle = \arg\min_{f_1 \cdots f_n} \sum_{i=1}^{N} L(\sum_{j=1}^{n} P(C_j|q_i) f_j(X_i, \omega_j), y_i) \tag{4}$$

where n is the number of identified ranking-sensitive query topics; $P(C_j|q_i)$ represents the probability that q_i belongs to C_j; and ω_j denotes unknown parameters of the ranking function f_j corresponding to the query topic C_j. Intuitively, if query q_i is highly correlated to a query topic C_j, i.e. with high value of $P(C_j|q_i)$, the loss of ranking under q_i will be much associated with learning ω_j.

3.3.2 Topical RankSVM

The learning task in Eq. 4 is specified when the form of the ranking function f and that of the loss function L are defined, for example, we can use linear function as ranking function, i.e. $f(X, \omega) = \omega^T X$, and L_2 norm as loss function, i.e. $L(f(X, \omega), y) = \|f(X, \omega) - y\|^2$. The ranking specialization framework is quite general and flexible in the sense that it can apply different ranking algorithms. In this paper, we use RankSVM as an example to demonstrate its advantages. For simplicity, we consider the linear function $f(X, \omega) = \omega^T X$. We refer to our method as *Topical RankSVM*. Note that the same idea can also be applied to

other ranking algorithms, such as RankNet and RankBoost, by modifying the respective loss function according to Eq. 4.

• **RankSVM:**

We first make a review of RankSVM [10, 12]. Its learning task is defined as the following quadratic programming problem:

$$\begin{aligned} &\min_{\omega,\xi_{q,i,j}} \frac{1}{2}\|\omega\|^2 + c\sum_{q,i,j}\xi_{q,i,j} \\ &s.t.\ \omega^T X_i^q \geq \omega^T X_j^q + 1 - \xi_{q,i,j}, \\ &\forall X_i^q \succ X_j^q,\ \xi_{q,i,j} \geq 0 \end{aligned} \tag{5}$$

where $X_i^q \succ X_j^q$ implies that document i is ranked ahead of j with respect to query q in the training dataset; $\xi_{q,i,j}$ denotes slack variable; and $\|\omega\|^2$ represents structural loss.

• **Topical RankSVM:**

We then describe *Topical RankSVM*. Inspired by the fact that the ranking model of one specific ranking-sensitive query topic depends more on the training query-document pairs, the query in which has higher correlation to the certain query topic, we modify Eq. 5 into the optimization problem of *Topical RankSVM*:

$$\begin{aligned} &\min_{\omega,\xi_{q,i,j}} \frac{1}{2}\sum_{k=1}^{n}\|\omega_k\|^2 + c\sum_{q,i,j}\xi_{q,i,j} \\ &s.t.\ \sum_{k=1}^{n} P(C_k|q)\omega_k^T X_i^q \geq \sum_{k=1}^{n} P(C_k|q)\omega_k^T X_j^q + 1 - \xi_{q,i,j}, \\ &\forall X_i^q \succ X_j^q,\ \xi_{q,i,j} \geq 0 \end{aligned} \tag{6}$$

where ω_k denotes the parameters of ranking function with respect to query topic C_k. Note that we can use different feature sets for different query topics by using this method, but for simplicity, we didn't try it in this work. The optimization problem can be solved by employing existing optimization techniques, the computation details of which, though tedious, are rather standard and will not be presented here.

Note that, there are several advantages by using *Topical RankSVM*. Firstly, we are able to embed ranking-sensitive query topics directly into the ranking algorithm and learn multiple ranking functions for different query topics, simultaneously. Secondly, the global loss serves as a unified relevance target for all topic-specific ranking models, which can boost the relevance than training models separately. And, we employ all of the training queries to learn ranking models of each query category, avoiding the reduction of the training examples. Moreover, compared to previous work [9, 13], our approach is not less efficient on training time, which has been proved in the experiments on a large scale dataset. The same idea can also be applied on ranking algorithms other than RankSVM, in the similar way. In the rest of this section, we will employ a testing method which is consistent with the training method, in the sense that both of them focus on optimizing the same risk of ranking.

3.4 Ensemble Ranking for New Queries

After obtaining multiple ranking models corresponding to ranking-sensitive query topics, we employ an un-supervised ensemble method for improving ranking at query time. The intuition is that: Similar queries in ranking-sensitive feature space are more likely to hold similar ranking characteristics, therefore, if one query topic has higher correlation to the testing query, the corresponding ranking models should contribute more to the final ranking of the testing query.

Assuming that $\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n$ are ranking models learned with the method in 3.3 and correspond to query topics C_1, C_2, $\cdots$, C_n respectively; $\tilde{q}$ is a testing query, and $\{\tilde{d}_1, \tilde{d}_2, \cdots, \tilde{d}_{M_{\tilde{q}}}\}$ is the set of documents to be ranked with respect to $\tilde{q}$; and $P(C_1|\tilde{q})$, $P(C_2|\tilde{q})$, $\cdots$, $P(C_n|\tilde{q})$ are the probabilities that the new testing query belongs to query topics. Then, we compute the ranking score $\mathcal{S}(\tilde{q}, \tilde{d}_i)$ for each document $\tilde{d}_i$ $(i = 1, \cdots, M_{\tilde{q}})$ as follows:

$$\mathcal{S}(\tilde{q}, \tilde{d}_i) = \sum_{k=1}^{n} P(C_k|\tilde{q})\hat{f}_k(\mathbf{x}^{\tilde{q}\tilde{d}_i}, \omega_k) \tag{7}$$

where $\mathbf{x}^{\tilde{q}\tilde{d}_i}$ is the ranking feature vector of the query-document pair of $\tilde{q}$ and $\tilde{d}_i$; ω_k denotes parameters of $\hat{f}_k$. Then, we can obtain the final ranking of documents under $\tilde{q}$ according to the aggregated ranking scores computed by Eq. 7. Note that this testing approach tries to minimize the ranking risk consistent with that in training process.

4. EXPERIMENTAL SETUP

This section presents our evaluation setup. First, we describe the datasets we used in the experiments (Section 4.1). Then, we describe our evaluation metrics (Section 4.2) and the ranking methods to compare (Section 4.3) for the experimental results reported in Section 5.

4.1 Data Collection

In the experiment, we used three datasets, including both the publicly benchmark dataset and that obtained from a commercial search engine.

LETOR 3.0:

LETOR 3.0 [16] is a benchmark dataset for research on ranking [1]. We use TREC2003 and TREC2004 in LETOR 3.0 to evaluate the performance of exploring ranking-sensitive query topics for improving ranking. TREC2003 contains 350 queries and TREC2004 contains 225 ones. For each query, there are about 1,000 associated documents. Each query-document pair is given a binary judgment: *relevant* or *irrelevant*. In total, there are 64 features for each query-document pair, which can be referred to [16] for the details.

Both of these two tracks classify all the queries into three pre-defined categories, including topic distillation (TD), homepage finding (HP) and named page finding (NP), according to search intent. The statistics of queries for three categories in LETOR 3.0 can be found in [16]. Note that, this pre-defined hard categorization can also be used to improve ranking by applying the same method in Section 3.3. In our experiment, we will compare the effects on improving ranking by using ranking-sensitive query topics with that by using the pre-defined categorization.

LETOR 4.0:

LETOR4.0 is a new release of benchmark dataset for research on ranking [1]. It uses the web page collection (50M pages) and two query sets from Million Query track of TREC 2007 and TREC 2008, called MQ2007 and MQ2008 for short. There are about 1700 queries in MQ2007 with labeled documents and about 800 queries in MQ2008 with labeled documents. Each query-document pair is represented by a 46-dimensional feature vector. And we used the data with the setting of supervised ranking to evaluate the performance of our proposed ranking approach.

Commercial search engine dataset (SE-Dataset):

We also conduct experiment on a dataset obtained from a major commercial search engine. This dataset contains

71,810 training queries and 1,227,094 query-document pairs for training as well as 7,668 testing queries and 252,086 query-document pairs for testing. All the training and test queries are randomly sampled from the real user traffic to the search engine. Each query is associated with its retrieved documents, along with human judged labels that represent the degrees of relevance of those documents with respect to the queries. There are five levels of relevance: perfect, excellent, good, fair, and bad. Features for each query-document pair used in building the ranking functions can be roughly grouped into the following categories: text-matching features, link-based features, user-click features, query and page classification features. We denote this dataset as *SE-Dataset*.

This dataset classifies all the queries into four semantic domains, including autos domain ($\mathcal{D}_{\text{auto}}$), local domain ($\mathcal{D}_{\text{local}}$), product domain ($\mathcal{D}_{\text{product}}$), and travel domain ($\mathcal{D}_{\text{travel}}$). For each query, SE-Dataset provides the pre-computed similarity score between the query and each query domain. The value of the similarity score is in the range of $[0, 1]$, 0 meaning the smallest value of similarity while 1 meaning the largest value of similarity. There are some queries in SE-Data that have the 0 similarity scores with all of four pre-defined domains. In our experiment, we define a new soft query classification, which contains five classes, including $\mathcal{D}_{\text{auto}}$, $\mathcal{D}_{\text{local}}$, $\mathcal{D}_{\text{product}}$, $\mathcal{D}_{\text{travel}}$, and *general* domain $\mathcal{D}_{\text{general}}$. For each query q, we set the similarity score with respect to *general* domain class as 1, and after normalizing similarity scores with respect to all five classes, we can obtain a soft query classification.

4.2 Evaluation Metrics

We adapt the following information retrieval metrics to evaluate the performance of the ranking function.

•**Normalized Discounted Cumulative Gain (NDCG)**: NDCG has been widely used to assess relevance in the context of search engines [11]. For a ranked list of documents, the NDCG score at position n is calculated as follows:

$$\text{NDCG}(n) \equiv Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{\log_2(j+1)} \tag{8}$$

where j is the position in the document list, $r(j)$ is the rating of the j-th document in the list, and Z_n is the normalization factor which is chosen so that the perfect list gets a NDCG score of 1.

•**Mean Average of Precision(MAP)**: Average precision for each query is defined as the mean of the precision at n values calculated after each relevant documents was retrieved. The final MAP value is defined as the mean of average precisions of all queries in the test set. This metrics is the most commonly used single-value summary of a run over a set of queries. Thus, MAP is calculated as:

$$MAP = \frac{1}{|Q|} \sum_{q \in Qr} \frac{\sum_{n=1}^{N} (P@n \times rel(n))}{|R_q|}$$

where Q is a set of test queries, R_q is the set of relevant document for q, n is the rank position, N is the number of retrieved documents, $rel()$ is a binary function on the relevance of a given rank.

4.3 Ranking Methods Compared

To evaluate the performance of our approach employing ranking-sensitive query topics in learning the ranking function, we compare the performance of the following methods:

1. RSVM: As a baseline method, we employ RankSVM (denoted as RSVM) and conduct training over all the training queries to learn one single ranking model. At testing time, we use the single model to generate ranking results for all testing queries.

2. CRSVM: In this method, we apply the pre-defined query categorization into our proposed SVM-based learning method, where each query category is viewed as one query topic. In LETOR 3.0 dataset, each query can only belong to only one category. Thus, this method equals to separating the training queries based on query categorization and training a specialized model for each category of queries. In commercial search engine dataset, we can employ the pre-defined soft query categorization into our proposed SVM-based learning method directly. We call this method "*Semantic-class based RankSVM*", denoted as CRSVM. At testing time, for hard query categorization, we choose the ranking model according to the category of each testing query; for soft query categorization, we apply the ensemble approach in Section 3.4 to generate ranking results.

3. LRSVM: In this method, we simulate the general idea in [9]. After identifying the ranking-sensitive query categories, we classify each training query into the closest query category. Based on this hard partition of training queries, we train separate ranking model for each query category using its own fraction of training queries. This method is usually called "*Local Ranking*" in previous work. By using RankSVM, we denote this method as LRSVM. At testing time, according to the correlation between the test query and query categories, we select the ranking model of the most correlated query category and use it to generate the ranking results.

4. TRSVM: In this method, after identifying the ranking-sensitive query topics (Section 3.2), we employ the proposed "*Topical RankSVM*" (Section 3.3) with topic distribution of training queries to learn ranking models for those query topics. At testing time, we use the proposed ensemble method (Section 3.4) to generate the ranking results for testing queries.

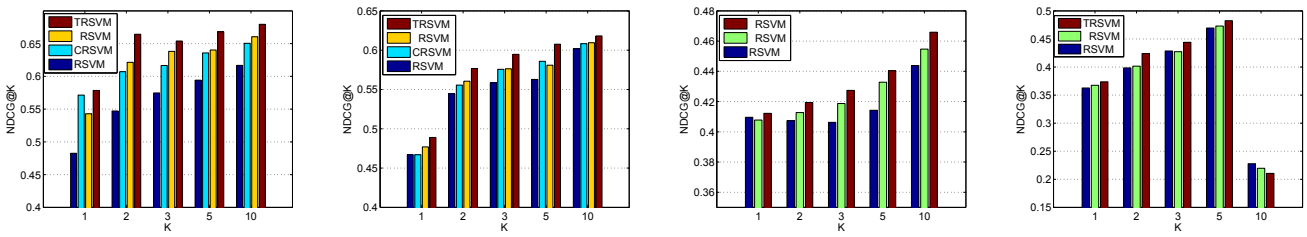
5. EXPERIMENTAL RESULTS

5.1 Experiments with LETOR Dataset

We evaluate the performance of the ranking methods on TREC2003 and TREC2004 in LETOR 3.0 as well as MQ2007 and MQ2008 in LETOR 4.0. For each of these datasets, we conduct 5-fold cross-validation experiments, using the default partitions in LETOR. For TREC2003 and TREC2004, since the data of each class (TD, NP, HP) is split into 5 folds, we combine respective i-th fold ($i = 1, \cdots, 5$) of each class together to form up the i-th fold of the whole dataset.

To identify ranking-sensitive query topics, we use BM25 as the reference model to rank documents and choose the top $T = 50$ ranked documents (if the total number of documents under one query is lower than T, all documents will be used) as pseudo feedback to create ranking-sensitive features.

The topic number n is crucial to the performance of TRSVM and LRSVM. Since there are three pre-defined classes in TREC2003 and TREC2004, we select the value as $n = 3$ to compare the performance of TRSVM with that of LRSVM and CRSVM for experiments on TREC2003 and TREC2004. Since there is no pre-defined class in MQ2007 and MQ2008, we will not test the performance of CRSVM on these two dataset. And, we set $n = 10$ to compare the performance of TRSVM with that of LRSVM for experiments on MQ2007



(a) TREC2003 (b) TREC2004 (c) MQ2007 (d) MQ2008

Figure 2: Ranking relevance in terms of NDCG@K of TRSVM compared with other methods on LETOR 3.0 and 4.0.

and MQ2008. In practice, this parameter is tuned automatically based on a validation set. In order to clearly illustrate the influence of this parameter on the ranking performance, we will also present the results with respect to different values of the topic/category number.

For RSVM, we can make use of its results provided in LETOR. For all the SVM models in the experiment, we employ the linear SVM. This is because the LETOR data set offers results of linear RankSVM.

5.1.1 Relevance Comparisons

The purpose of this experiment is to compare the average relevance of different ranking algorithms on different benchmark datasets. Figure 2(a) and 2(b) illustrate the NDCG values of TRSVM compared with RSVM, CRSVM and LRSVM on TREC2003 and TREC2004, respectively. From these two figures, we observe that, by building different ranking models with respect to different query categories/topics, TRSVM, CRSVM and LRSVM out-perform the single model learned by RSVM on both dataset. Furthermore, by extracting the ranking-sensitive query topics and applying the proposed unified learning method, TRSVM give better relevance than CRSVM and LRSVM. We conduct t-test on the improvements in terms of NDCG@3, and the results indicate that for both TREC2003 and TREC2004, the improvements of TRSVM over other ranking methods are statistically significant (p-value$<$ 0.05).

In Table 1, we report the MAP scores of TRSVM compared with RSVM, CRSVM, LRSVM on TREC2003 and TREC2004, respectively. Table 1 indicates that TRSVM achieves much better relevance than RSVM, CRSVM and LRSVM. In particular, TRSVM achieves a gain of about 9% relative to RSVM on TREC2003 as well as 6% relative to RSVM on TREC2004; and TRSVM obtains more gains than CRSVM and LRSVM on both dataset.

Table 1: MAP value of RSVM, CRSVM, LRSVM, and TRSVM on TREC2003 and TREC2004

Ranking Method	TREC2003	Gain	TREC2004	Gain
RSVM	0.578	-	0.501	-
CRSVM	0.601	+4%	0.513	+2%
LRSVM	0.605	+5%	0.521	+4%
TRSVM	0.628	+9%	0.532	+6%

Moreover, Figure 2(c) and 2(d) illustrate the NDCG values of TRSVM compared with RSVM, and LRSVM on MQ2007 and MQ2008, respectively. From these two figures, we also observe that, on both dataset TRSVM and LRSVM outperform the single model learned by RSVM by employing different ranking models with respect to different query categories/topics. Furthermore, TRSVM, which extracts the ranking-sensitive query topics and applying the proposed unified learning method, can reach better ranking relevance than LRSVM based on local ranking. (Note that the NDCG@10 of queries in MQ2008 is much lower because a portion of queries have less than 10 documents to rank in this dataset.) We conduct t-test on the improvements in terms of NDCG@3, and find that the improvements of TRSVM over other ranking methods are statistically significant (p-value$<$ 0.05).

Table 2: MAP value of RSVM, LRSVM, and TRSVM on MQ2003 and MQ2004

Ranking Method	MQ2007	Gain	MQ2008	Gain
RSVM	0.464	-	0.470	-
LRSVM	0.474	+2%	0.477	+1%
TRSVM	0.481	+4%	0.489	+4%

Table 2 demonstrates the MAP value of TRSVM compared with RSVM as well as LRSVM on MQ2007 and MQ2008, respectively. From the table, we can observe that TRSVM achieves much better relevance than RSVM and LRSVM. In particular, TRSVM achieves a gain of about 4% relative to RSVM on both MQ2007 and MQ2008; and TRSVM obtains more gain than LRSVM on both datasets.

5.1.2 Results Analysis

In the following, we will investigate the reason that TRSVM can achieve better ranking relevance than the single model RSVM, the class-based ranking approach CRSVM, and the local ranking approach, LRSVM.

I. Multiple ranking models v.s. single model

Table 3 and 4 demonstrates the 8 most important features for single model learned by RSVM and for separate models corresponding to query topics learned by CRSVM and TRSVM, respectively. These results are based on the experiment on TREC2003. The feature importance is measured by the absolute value of learned feature weight. The detailed description of features can be found in [16].

From these tables, we can observe that, introducing query difference in terms of query classification/clustering into ranking can help to build multiple ranking models with respect to various query classes (CRSVM) or query topics (TRSVM). These ranking models can represent multiple fine-grained ranking characteristics of various query classes/topics, while the single ranking model can only describe a coarse summarization of ranking characteristics over all various queries.

In particular, if we build one single ranking model using RSVM, the top 5 most important features of the model are *weighted in-link*, *weighted out-link*, *TF-IDF of title*, *TF of title*, and *TF of body*, as shown in Table 3. To verify whether these five features are most important to ranking for most of queries, we conduct an experiment as follows: we randomly sample 20 queries from the whole dataset and build respective ranking models for each query based on the documents and labels associated with the certain query, then we compute the respective feature importance of each query. We randomly sample 20 queries for 3 times, and there are only in average 6.3 queries (31.5%) whose top 5 most important features include at least three of top 5 most important features of the model learned by RSVM.

To gain an understanding of what is the better way to use

Table 3: Top 10 most important features for RSVM and CRSVM on TREC2003

RSVM	CRSVM (TD)	CRSVM (NP)	CRSVM (HP)
weighted in-link	sitemap based score propagation	BM25 of whole document	number of slash in URL
weighted out-link	sitemap based term propagation	LMIR.JM of whole document	HostRank
TF-IDF of title	DL of title	sitemap based score propagation	HITS hub
TF of title	length of URL	sitemap based term propagation	DL of URL
TF of body	HostRank	LMIR.JM of anchor	length of URL
TF-IDF of whole document	BM25 of title	BM25 of anchor	LMIR.JM of title
length of URL	DL of URL	LMIR.DIR of whole document	sitemap based score propagation
topical PageRank	BM25 of anchor	LMIR.ABS of whole document	sitemap based term propagation

Table 4: Top 10 most important features for TRSVM on TREC2003

TRSVM (topic-1)	TRSVM (topic-2)	TRSVM (topic-3)
sitemap based term propagation	number of slash in URL	length of URL
sitemap based score propagation	HostRank	outlink number
length of URL	HITS sub	sitemap based term propagation
number of slash in URL	sitemap based score propagation	sitemap based score propagation
DL or URL	sitemap based term propagation	number of slash in URL
weighted in-link	uniform out-link	HITS sub
number of child page	Outlink number	DL or URL
BM25 of title	LMIR.ABS of URL	DL or title

the information of ranking-sensitive query topics to improve ranking, we perform a study on the effects of adding query topics information as features in learning. Specifically, for each query-document pair in both training and testing set, we use the query's topic distribution as additional features and apply RSVM to learn the single ranking model on expanded feature space. The testing results show that this method does not increase the ranking performance significantly, but even performs worse than the multiple ranking approach. The similar experiment on SE-Dataset also reports the same observation.

II. Ranking-sensitive query topics v.s. pre-defined semantic classes

After considering query different in terms of query classification(CRSVM)/clustering(TRSVM), we can build multiple ranking models to represent different fine-grained ranking characteristics. For example, by using CRSVM, we can learn three ranking models corresponding to three query classes: topic distillation (TD), namepage finding (NP), and homepage finding (HP). Table 3 shows the top 8 most important features of each learned model. From this table, we can find that each of the three models learned by CRSVM are quite different with that of RSVM. In particular, some features, such as *TF-IDF of title*, *weighted out-link* etc., are important for RSVM, but are not essential for ranking of each specific query class. Furthermore, the most important features of each query class are diverse, for example, *sitemap based score/term propagation* are most important to TD queries; NP queries depend mostly on language model based (LMIR) and probabilistic (BM25) features; and *number of slash in URL* is the most important one for HP queries but not included in top 8 for the other two classes. Similarly, Table 4 shows that three ranking models learned by TRSVM are different with not only the single model of RSVM but also multiple models learned by CRSVM.

To verify whether TRSVM can obtain multiple ranking models that better represent ranking characteristics of respective query topics than other methods, we conduct the following experiment: for each identified query topic, we randomly sample 20 queries from those belonging to this query topic, and build respective ranking model for each of these 20 query based on their own associated documents and labels; then, we compute the respective feature importance of these queries and validate whether they include the top important features learned by TRSVM. We conduct this experiment on 5 query topics, and randomly sample 20 queries under each topic for 3 times. There are in average 14.8 queries (74%) whose top 5 most important features include at least three of those of the model learned by TRSVM. For LRSVM and CRSVM, the results shows that there are in average 12.7 (63.5%) and 11.3(56.5%) queries whose top 5 most important features include at least three of those of the model learned by LRSVM and CRSVM, respectively.

III. Why TRSVM performs better?

We hypothesize the reasons of why TRSVM can perform better than LRSVM and CRSVM as follows. Firstly, instead of pre-defined query classification, TRSVM and LRSVM represent each query by aggregating the ranking features of documents under such query and conduct un-supervised clustering method to identify ranking-sensitive query topics, which can be better to distinguish queries based on their various ranking characteristics.

Secondly, LRSVM and CRSVM employ hard query classification/clustering and build multiple ranking models corresponding to each query class/cluster, however, many queries can fit into more than one classes/clusters, therefore, TRSVM can benefit ranking performance by taking advantage of soft query categorization into building multiple ranking models.

Another advantage of TRSVM is avoidance of the reduction in the number of training examples. Specifically, either LRSVM or CRSVM uses only a part of training dataset to learn the ranking model for each query class/cluster. It may cause the declining accuracy due to the lack of enough training examples. However, TRSVM can avoid the reduction of training dataset size since it uses all training examples, with different weights based on query soft clustering, in learning the ranking model of each query topic.

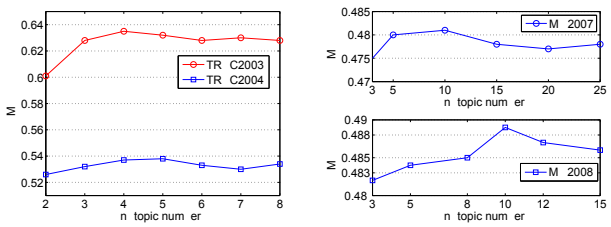
5.1.3 *Effects of Different Parameter Settings*

In this experiment, we explore the effects of different settings of the parameter n, i.e. the number of query topics, on ranking performance by conducting comparison study with varying the value of n.

Figure 3 show the performance of TRSVM on Letor dataset with varying values of n in terms of MAP. From the figure, we can find that, as n increases, the performances first increase, but then as n becomes much larger, there is no significant raising on the performance, and the performance is even deteriorated at some time. More specifically:

•When setting only a small number of query topics, the performances of TRSVM are not so good since the identified query topics are a bit broad to reflect the query difference.

•As setting the higher number of query topics, we can im-



(a) TREC2003 and TREC2004 (b) MQ2007 and MQ2008

Figure 3: Ranking performance (MAP) of TRSVM on the Letor datasets against varying topic number n

prove the performances since identified query topics are more fine-grained to reflect query difference in ranking.

•In the ideal case, with increasing number of query topics, the proposed method can achieve much better ranking performance, since we can build ranking models, each of which focuses on more fine-grained ranking characteristics. However, the actual results indicates that the ranking performances do not increase significantly but even deteriorate at some time. We hypothesize that, the extracted query features and identified ranking-sensitive query topics, though more effective than other query categorization method for enhancing ranking, are still not optimal for recognizing the ranking-sensitive query topics; therefore, when each identified query topic becomes more fine-grained, the bias of query categorization will be enlarged so as to deteriorate the ranking performance.

5.2 Experiments with SE-Dataset

We compare the performance of proposed ranking methods (TRSVM) with the baselines of the single model approach (RSVM), class-based approach (CRSVM) and the local ranking based approach (LRSVM) on the commercial search engine dataset (SE-Dataset). The pre-defined classes in CRSVM include *auto*, *local*, *product*, *travel*, *general* as described in Section 4.1.

To identify ranking-sensitive query topics, we use BM25 as the reference model to rank documents and choose the top $T = 20$ documents (if the total number of documents under one query is lower than T, all documents will be used) as pseudo feedback to create ranking-sensitive features.

We set the query topic/cluster number in TRSVM/LRSVM, i.e. the parameter n, as $n = 20$ in the experiment. In practice, this parameter is tuned automatically based on a validation set. In order to clearly illustrate the influence of this parameter on the ranking performance, we will also present the results with respect to different values of the topic/category number.

5.2.1 *Performance Comparisons*

Figure 4 demonstrates the NDCG values of TRSVM compared with RSVM, CRSVM, and LRSVM on SE-Dataset. From the figures, we observe that, by building different ranking models with respect to different query categories/topics, TRSVM, LRSVM and CRSVM out-perform the single model learned by RSVM. Furthermore, by extracting the ranking-sensitive query topics and applying the proposed unified learning method, TRSVM give better performance than CRSVM and LRSVM. We conduct t-tests on the improvements in terms of NDCG@3, the results of which indicate that the improvements of TRSVM over RSVM, CRSVM and LRSVM are statistically significant (p-value$<$ 0.05).

5.2.2 *Effects of Different Aggregation for Query Features*

In this experiment, we test the performance of the proposed TRSVM method with using different information for aggregating query features. By default, we compute the mean values of ranking features of top pseudo feedbacks as the feature vector of the query. In this experiment, we explore the effects of adding extra statistical quantities beyond means into the query feature vector. We use variance as the extra statistical quantity in our experiment. The ranking method using the expanded query features is denoted as TRSVM$_{var}$. Moreover, we test the effects of making use of the knowledge of ranking feature importance into the aggregation for query features. In particular, we first learn the ranking function by using a sample of the training dataset. After that, we can obtain the importance of each feature for learning the ranking function. Then, we incorporate feature importance as weight into computing query features. We denote the ranking method with incorporated feature importance as TRSVM$_{impt}$. We also test the ranking method which both expands query features with aggregated variance values and incorporates feature importance in identifying query topics, which is denoted as TRSVM$_{var-impt}$.

Figure 5 shows the performances of the proposed TRSVM method with applying various information into the aggregation for query features. From this figure, we observe that, utilizing feature importance into identifying query topics can increase the ranking performance. After conducting t-test on the improvement in terms of NDCG@3, we find that TRSVM$_{impt}$ out-perform TRSVM with p-value less than 0.02. Figure 5 also illustrates that expanding query features with aggregated variance value (TRSVM$_{var}$) does not improve ranking performance but even cause a decreased accuracy than TRSVM, the reason of which could be variance is not useful to identify ranking-sensitive query topics. However, it does not indicate that other statistical quantities are not useful for query topic recognition either. It would be an interesting future work to explore what kind of statistical quantities are essential to identify ranking-sensitive query topics. Moreover, Figure 5 demonstrates that TRSVM$_{var-impt}$ can achieve better performance than TRSVM$_{var}$ and TRSVM, but not TRSVM$_{impt}$, which also indicate that using feature importance can boost ranking performance while expanding query features with aggregated variance may not be useful for improve ranking.

5.2.3 *Robustness to Noisy Query Topics*

In the following, we perform experiments to evaluate the robustness of our divide-and-conquer ranking approach to noisy ranking-sensitive query topics. We manually add some noises into the ranking-sensitive query topics and test this effects on the overall ranking relevance. Specifically, after identifying ranking-sensitive query topics, we randomly select a portion of training queries and change their topic distribution into random values. Then, we employ TRSVM with these noisy ranking-sensitive query topics.

Figure 6 illustrates the NDCG@1,3,5 scores for the testing queries (on MQ2007 and MQ2008) against varying amount of queries with noisy topic distribution in the training data. The figures show that, although the relevance declines as the amount of queries with noisy topic distribution increases, TRSVM even outperform single ranking model approach RSVM and therefore is robust to the noisy ranking-sensitive query topics. And we can find the similar results when we take the same experiments on LETOR 3.0 dataset.

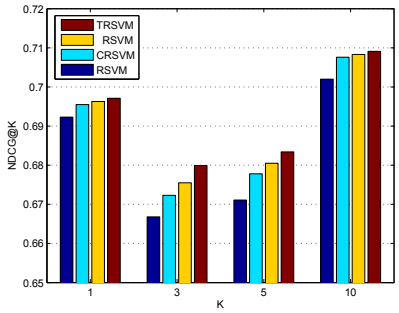


Figure 4: Relevance (ndcg@k) of TRSVM compared with the other methods on SE-Dataset

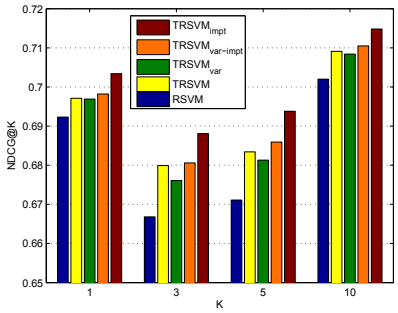


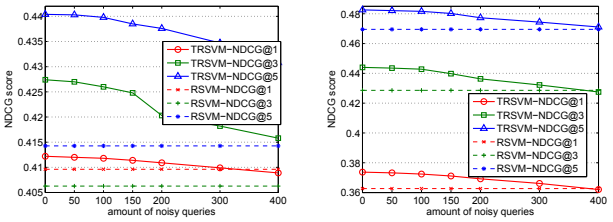
Figure 5: Relevance (ndcg@k) of TRSVM with different aggregation method for query features on SE-Dataset

The robustness of our divide-and-conquer ranking approach can also be demonstrated on SE-Dataset, especially from Figure 5. This figure shows that different aggregation methods give rise to different ranking-sensitive query topics, and hence different ranking performance; but, in spite of such difference, any of these different query categorizations can be used to trained the ranking models which outperform the single model approach (RSVM). Therefore, the proposed divide-and-conquer framework is robust to the noisy ranking-sensitive query topics.

6. CONCLUSION AND FUTURE WORK

In this paper, we point out that, due to great variance of queries and different ranking characteristics of Web queries, single ranking model is not appropriate for diverse types of queries. We employ a divide-and-conquer approach to learn multiple ranking functions according to diverse ranking characteristics of queries. We first identify ranking-sensitive query topics based on query features from pseudo feedbacks and prior knowledge of feature importance. Then, we propose a unified learning process to obtain ranking models corresponding to recognized query topics. An ensemble approach is applied to compute ranking for test queries by making use of multiple topic-specific ranking models. Experimental results illustrate that the proposed approach can significantly improve the ranking relevance over the single model approach and a straightforward local ranking approach, and the identified ranking-sensitive topics are more useful for improving ranking than pre-defined query categorization.

In future, we plan to explore deeply on what kind of features as well as which aggregation method are more essential to identify ranking-sensitive topics for queries. We also plan to investigate how to recognize the ranking-sensitive query topics jointly with learning the ranking function. We will also explore hierarchical query topics in the framework. Moreover, we intend to explore how to extend the current framework in order to allow incremental updating on any of ranking models without hurting the relevance of other queries.



(a) MQ2007 (b) MQ2008

Figure 6: Ranking performance (NDCG@K) of TRSVM against varying amount of noisy queries

7. REFERENCES

[1] Letor dataset website. http://research.microsoft.com/en-us/um/beijing/projects/letor/.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[3] S. Beitzel, E. Jensen, A. Chowdhury, and O. Frieder. Varying approaches to topical web query classification. In *Proc. of SIGIR*, 2007.

[4] B. Bolstad, R. Irizarry, M. Astrand, and T. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19:185–193, 2003.

[5] A. Broder. A taxonomy of web search. *SIGIR Forum*, 2002.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of ICML*, 2005.

[7] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proc. of ICML*, 2007.

[8] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Journal of JMLR*, 2003.

[9] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *Proc. of SIGIR*, 2008.

[10] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Proc. of ICANN*, 1999.

[11] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of ir techniques. In *ACM Transactions on Information Retrieval*, 2002.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of KDD*, 2002.

[13] I. Kang and G. Kim. Query type classification for web document retrieval. In *Proc. of SIGIR*, 2003.

[14] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proc. of SIGIR*, 2001.

[15] P. Li, B. Christopher, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proc. of NIPS*, 2007.

[16] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proc. of SIGIR*, 2007.

[17] S. Robertson. Overview of the okapi projects. In *Journal of Documentation*, 1998.

[18] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. In *Journal of ACM*, 1968.

[19] D. Shen, J. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proc. of SIGIR*, 2006.

[20] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63:411–423, 2000.

[21] E. Voorhees and D. Harman. Trec: Experiment and evaluation in information retrieval. In *MIT Press*, 2005.

[22] H. Zha, Z. Zheng, H. Fu, and G. Sun. Incorporating query difference for learning retrieval functions in information retrieval. In *Proc. CIKM*, 2006.

[23] Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgments. In *Proc. of SIGIR*, 2007.

[24] Z. Zheng, H. Zha, and G. Sun. Query-level learning to rank using isotonic regression. In *Proc. of the 46th Allerton Conf. on Comm., Control and Computing*, 2008.