Query Dependent Ranking Using K-Nearest Neighbor*

Xiubo Geng *
Institue of Computing Technology
Chinese Academy of Sciences
Beijing, 100190 P.R. China
rainy0211@gmail.com

Tie-Yan Liu
Microsoft Research Asia
No.49 Zhichun Road, Haidian District
Beijing 100190, P.R. China
tyliu@microsoft.com

Tao Qin *
Dept. Electronic Engineering
Tsinghua University
Beijing, 100084, P.R. China
tsintao@gmail.com

Andrew Arnold *
Machine Learning Department
Carnegie Mellon University
Pittsburgh,PA 15213, USA
aarnold@cs.cmu.edu

Hang Li
Microsoft Research Asia
No.49 Zhichun Road, Haidian District
Beijing 100190, P.R. China
hangli@microsoft.com

Heung-Yeung Shum
Microsoft Corporation
Redmond,WA 98115, USA
hshum@microsoft.com

ABSTRACT

Many ranking models have been proposed in information retrieval, and recently machine learning techniques have also been applied to ranking model construction. Most of the existing methods do not take into consideration the fact that significant differences exist between queries, and only resort to a single function in ranking of documents. In this paper, we argue that it is necessary to employ different ranking models for different queries and conduct what we call query-dependent ranking. As the first such attempt, we propose a K-Nearest Neighbor (KNN) method for query-dependent ranking. We first consider an online method which creates a ranking model for a given query by using the labeled neighbors of the query in the query feature space and then rank the documents with respect to the query using the created model. Next, we give two offline approximations of the method, which create the ranking models in advance to enhance the efficiency of ranking. And we prove a theory which indicates that the approximations are accurate in terms of difference in loss of prediction, if the learning algorithm used is stable with respect to minor changes in training examples. Our experimental results show that the proposed online and offline methods both outperform the baseline method of using a single ranking function.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Experimentation

*The work was done when the first, the third, and fourth authors were interns at Microsoft Research Asia.

SIGIR'08, July 20–24, 2008, Singapore.

Keywords

Query dependent ranking, *k*-nearest neighbor, stability

1. INTRODUCTION

Ranking will continually be an important research topic, as long as search and other information retrieval applications keep developing and growing. When used in search, ranking becomes a task as follows. Given a query, the documents related to the query in the document repository are sorted according to their relevance to the query using a ranking model, and a list of top ranked documents is presented to the user. The key problem for the related research is to develop a ranking model that best represents relevance.

Many models have been proposed for ranking, such as the Boolean model [2], the vector space model [23, 24], BM25 [21] and language model for IR [13, 18]. Recently, machine learning techniques called learning to rank have also been applied to automatic ranking model construction [5, 6, 7, 11, 17]. By leveraging labeled training data and machine learning algorithms, this approach is able to make the tuning of ranking model theoretically sounder and practically more effective. The training data consists of queries, their associated documents and labels representing relevance of documents. In this paper, we also base our work on learning to rank.

In most of the previous work, a single ranking function is used to handle all queries. This may not be appropriate, particularly for web search, as explained below. Instead, it would be better to exploit different ranking models for different queries. In this paper, we refer to this approach as query-dependent ranking. (We note that some authors use the term 'query dependent ranking' to refer to the document ranking process of using both query information and document information, in contrast to the process of using document information alone [20]. We use the term differently in this paper.)

Queries in web search may vary largely in semantics and the users' intensions they represent, in forms they appear, and in numbers of relevant documents they have in the document repository. For example, queries can be navigational, informational, or transactional [22]. Queries can be personal names, product names, or terminology. Queries can be phrases, combinations of phrases, or natural language sentences. Queries can be short or long. Queries can be popular (which have many relevant documents) or rare (which only have a few relevant documents). Using a single model alone would make compromises among the cases and result in lower accuracy in relevance ranking.

The IR community has realized the necessity of conducting query-dependent ranking. However, efforts were mainly made on query classification [3, 4, 12, 14, 22, 25, 26], but not ranking model construction or ranking model learning. The only exception is Kang and Kim's work [12], to our knowledge, in which queries were classified into two categories based on search intension and two different ranking models were tuned and used for the two categories. Therefore, more investigations on the approach are needed, which is also the motivation of this work.

Inspired by previous work [8], we propose a query-dependent method for ranking model construction, on the basis of K-Nearest Neighbor (KNN). We position the training queries into the query feature space in which each query is represented by a point. In ranking, given a test query we retrieve its k nearest training queries, learn a ranking model with these training queries, and then rank the documents associated with the test query using that model. The accuracy of ranking can be enhanced by employing the proposed method, due to the following reasons. First, in the method ranking for a query is conducted by leveraging the useful information of the similar queries and avoiding the negative effects from the dissimilar ones. Second, 'soft' classification of queries is carried out and similar queries are selected dynamically. Our experimental study has verified the superiority of the KNN method to both the single model approach and the query classification based approach.

Since KNN needs to conduct online training of the ranking model for each test query, and this would not be affordable in practice, we further propose two approximations of the method, which move the training offline. We give both theoretical justification and empirical verification for the two offline methods. Specifically, we prove that the approximations are accurate in terms of difference in loss of prediction, if the learning algorithm used is stable with respect to minor changes in training examples. The contributions of this paper include the following points:

(1) Proposal of the approach of query-dependent ranking,

(2) Development of KNN methods for query-dependent ranking,

(3) Theoretical and empirical investigations of the methods.

The rest of this paper is organized as follows. In Section 2, related work is presented. In Section 3, the KNN method for query-dependent ranking is proposed. In Section 4, experimental results are reported. In the last section, conclusions are made and future work is discussed.

2. RELATED WORK

There has not been much previous work on query dependent ranking, except [12], as explained above. The most relevant research topics are query classification and learning to rank.

Many efforts have been made on query classification. In [3, 4, 25], queries were classified according to topics, for instance, Computers, Entertainment, Information, etc. as used in KDD Cup 2005. In [12, 14, 22, 26], queries were classified according to users' search needs, for instance, topic distillation, named page finding, and homepage finding. Machine learning methods such as support vector machines were usually employed in the classification. However, query classification was not extensively applied to query dependent ranking, probably due to the difficulty of the query classification problem.

Recently, a large number of studies have been conducted on learning to rank and its application to information retrieval. Existing methods for learning to rank fall into three categories: the pointwise approach [17], which transforms ranking to classification or regression on single documents; the pair-wise approach [5, 7, 11], which formalizes ranking as classification on document pairs; and the list-wise approach [6, 28, 29], which directly minimizes a loss function defined on document lists. The KNN based method proposed in this paper can also be viewed as a new learning to rank method.

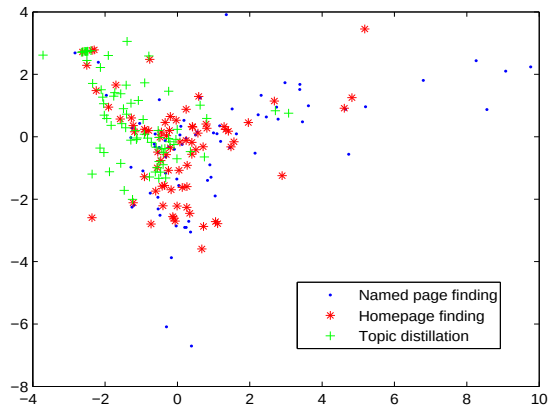


Figure 1: Distribution of Queries in TREC 2004 Data.

3. RANKING USING K-NEAREST NEIGHBOR

3.1 Motivation

Queries can vary greatly between each other in several perspectives, as explained above. For example, popular queries like 'soccer' can have many relevant documents and thus features measuring document popularity (e.g. PageRank) will be important for ranking the documents related to this query. In contrast, rare queries like 'SIGIR 2007 workshop on learning to rank' may only have a few relevant documents, and thus the use of document popularity in ranking is not even necessary. Using a single ranking model alone would not be able to deal with these cases properly. Naturally we think about employing the 'query dependent approach', in which we train and use different ranking models for different queries.

A straightforward approach to query dependent ranking would be to employ a 'hard' classification approach in which we classify queries into categories and train a ranking model for each category. We think, however, that it could be very difficult to achieve high performance with this approach .

When looking at the data, we often observe that it is hard to draw clear boundaries between the queries in different categories. Let us take the TREC 2004 web track data as an example. There are in total 225 queries in the dataset, which have been manually classified into three categories: topic distillation, named page finding, and homepage finding. The queries are also associated with documents and relevance labels of these documents.

We define features of queries, following the proposal in [26], and represent the queries in a 27-dimensional query feature space. We next reduce the space to 2-dimensions by using Principal Component Analysis (PCA). We then plot the queries in this reduced space and obtain the graph in Figure 1.

One can see from the figure that queries in different categories are mixed together and cannot be separated using hard classification boundaries. At the same time, one can observe that with high

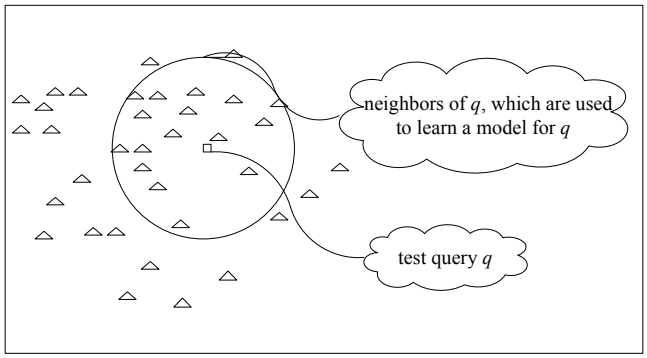


Figure 2: Illustration of KNN Online.

probability a query belongs to the same category as those of its neighbors. We refer to this observation as the 'locality property of queries'. This locality property motivates us to tackle the problem of query-dependent ranking using a K-Nearest Neighbor (KNN) approach. In some sense, we can view KNN as an algorithm performing 'soft' classification in the query feature space.

3.2 Online Method

3.2.1 KNN Online

We employ a K-Nearest Neighbor method for query dependent ranking. For each training query q_i(with corresponding training data as $S_{q_i}, i = 1, ..., m$) [1], we define a feature vector and represent it in the query feature space (a Euclidean space). Given a new test query q, we try to find the k closest training queries to it in terms of Euclidean distance. We then train a *local* ranking model online using the neighboring training queries (denoted as $N_k(q)$) and rank the documents of the test query using the trained local model. For local model training, we can in principle employ any existing learning to rank algorithm. In this paper, we choose Ranking SVM [11].

We call the corresponding algorithm 'KNN Online'. Figure 2 illustrates the workings of the algorithm where the square denotes test query q, triangles denote training queries, and the large circle denotes the neighborhood of query q. The details of the algorithm are presented in Figure 3.

Needless to say, the query features used in the method are critical to its accuracy. In this paper, we simply use the following heuristic method to derive query features and leave further investigation of the issue to future work.

For each query q, we use a reference model (in this paper BM25) to find its top T ranked documents, and take the mean of the feature values of the T documents as a feature of the query. For example, if one feature of the document is *tf-idf*, then the corresponding query feature becomes average *tf-idf* of top T ranked documents of the query. If there are many relevant documents, then it is very likely that the value of average *tf-idf* would be high.

3.2.2 Time Complexity

The time complexity of KNN Online is high, and most of the computation comes from step (c) (online training) and step (b) (finding k nearest neighbors).

At step (c), it trains the ranking model online from $S_{N_k(q)} \triangleq \cup_{q' \in N_k(q)} S_{q'}$. Usually model training is time consuming. For instance, the time complexity of training a Ranking SVM model is of polynomial order in number of document pairs [10].

At step (b), it finds k nearest neighbors in the query feature space. If we use a straightforward search algorithm, the time complexity will be of order $O(m \log m)$, where m is the number of training queries.

To reduce the aforementioned complexity, we further propose two algorithms, which move the time-consuming steps offline.

[1] S_{q_i} contains query q_i, the training instances derived from its associated documents and the relevance judgments. When we use Ranking SVM as the learning algorithm, S_{q_i} contains all the document pairs associated with training query q_i.

Algorithm: KNN Online

Input:

(1) A test query q and the associated documents to be ranked.

(2) Training data $\{S_{q_i}, i = 1, ..., m\}$.

(3) Reference model h_r (currently BM25).

(4) Number of nearest neighbors k.

Output: Ranked list for query q.

Algorithm:

Offline pre-processing:
For each training query q_i, use reference model h_r to find its top T ranked documents, and compute its query features from these documents.

Online training and testing:

(a) Use reference model h_r to find the top T ranked documents for query q, and compute q's query dependent features from these documents.

(b) Within the training data find k nearest neighbors of q, denoted as $N_k(q)$, with distance computed in the query feature space.

(c) Use the training set $S_{N_k(q)} \triangleq \cup_{q' \in N_k(q)} S_{q'}$ to learn a local model h_q.

(d) Apply h_q to the documents associated with query q, and obtain the ranked list.

Figure 3: KNN Online Algorithm.

3.3 Offline Methods

3.3.1 KNN Offline-1

To improve the efficiency of KNN Online, we consider a new algorithm in which we move the model training offline. We refer to this new algorithm as KNN Offline-1.

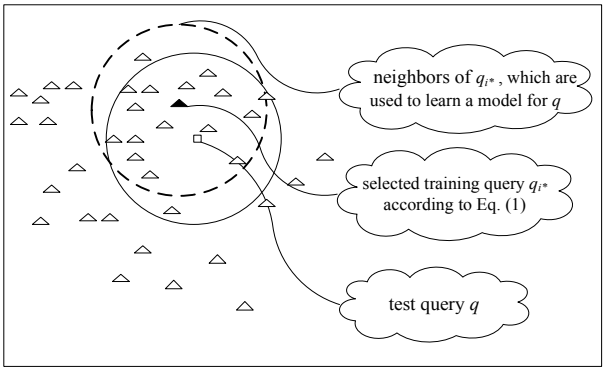


Figure 4: Illustration of KNN Offline-1.

Algorithm: KNN Offline-1

Input:

(1) A test query q and the associated documents to be ranked.

(2) Training data $\{S_{q_i}, i = 1, ..., m\}$.

(3) Reference model h_r (currently BM25).

(4) Number of nearest neighbors k.

Output: Ranked list for query q.

Algorithm:

Offline training:

(1) For each training query q_i, use reference model h_r to retrieve its top T documents, and compute its query features.

(2) For each training query q_i, find k nearest neighbors of q_i, denoted as $N_k(q_i)$, in the training data in the query feature space, and use training set $S_{N_k(q_i)}$ to learn a local model h_{q_i}.

Online testing:

(a) Use reference model h_r to find top T documents for query q, and compute its query features.

(b) Find k nearest neighbors of q, denoted as $N_k(q)$ in the training data in the query feature space.

(c) Find the most similar training set $S_{N_k(q_{i*})}$ by using Eq.(1).

(d) Apply $h_{q_{i*}}$ to the documents associated with query q, and obtain the ranked list.

Figure 5: KNN Offline-1 Algorithm.

KNN Offline-1 is a method as follows. First, for each training query q_i, we find its k nearest neighbors $N_k(q_i)$ in the query feature space. Then, we train a model h_{q_i} from $S_{N_k(q_i)}$ offline and in advance. In testing, for a new query q, we also find its k nearest neighbors $N_k(q)$. Then, we compare $S_{N_k(q)}$ with every $S_{N_k(q_i)}, i = 1, ..., m$ so as to find the one sharing the largest number of instances with $S_{N_k(q)}$.

$$S_{N_k(q_{i*})} = \arg\max_{S_{N_k(q_i)}} |S_{N_k(q_i)} \cap S_{N_k(q)}| \quad (1)$$

where $|.|$ denotes the number of instances in a set. Next, we use $h_{q_{i*}}$, the model of the selected training set (it has been created offline and in advance), to rank the documents of query q.

Figure 4 illustrates the workings of KNN Offline-1. Here the square represents the test query q and triangles represent the training queries. The triangles in the solid-line circle are the nearest neighbors of q, the solid triangle represents the selected training query q_{i*} according to Eq.(1), and the triangles in the dotted-line circle are the nearest neighbors of q_{i*}. In KNN Offline-1, the model learned from the triangles in the dotted-line circle is used to test query q. Figure 5 shows the details of the algorithm.

As will be discussed in Section 3.3.4, the model used in KNN Online and the model used in KNN Offline-1 are similar to each other, in terms of difference in loss of prediction.

3.3.2 KNN Offline-2

In KNN Offline-1, we can avoid online training. However, we introduce additional computation for searching for the most 'similar' training set. Furthermore, we still need to find the k nearest neighbors of the test query online which is also time-consuming.

Considering that online response time is critical for search engines, we propose a new algorithm, which we call KNN Offline-2, to further reduce the time complexity in applying KNN Offline-1.

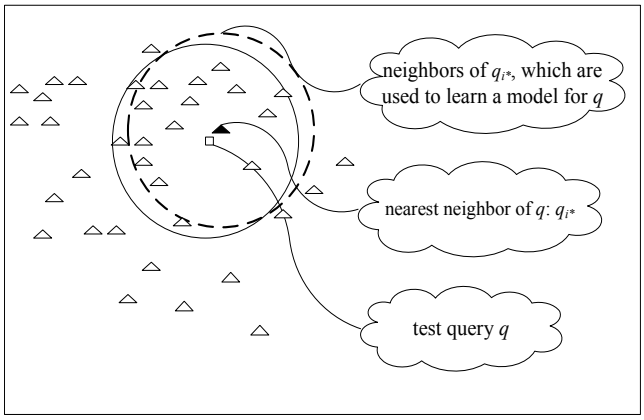


Figure 6: Illustration of KNN Offline-2.

Table 1: Time Complexities of Testing.

	KNN Offline-1	KNN Offline-2
Generate query feature	$O(n)$	$O(n)$
Find k nearest neighbors	$O(m \log k)$	$O(m)$
Find offline model	$O(mk)$	-
Rank	$O(n \log n)$	$O(n \log n)$

The idea of KNN Offline-2 is as follows. Instead of seeking the k nearest neighbors for the test query q, we only find its single nearest neighbor in the query feature space. Supposing that the nearest neighbor is q_{i*}, we directly apply the model $h_{q_{i*}}$ trained from $S_{N_k(q_{i*})}$ (offline and in advance) to test query q. In this way, we simplify the search of k nearest neighbors to that of a single nearest neighbor, and we no longer need to use Eq.(1) to find the most similar training set. As a result, the time complexity can be significantly reduced.

The basic idea of KNN Offline-2 is illustrated in Figure 6. As for the algorithm description, the only difference between KNN Offline-2 and KNN Offline-1 is that in the former steps (b) and (c) become 'Find the nearest neighbor of q, denoted as q_{i*}'. We omit the algorithm details.

3.3.3 Time Complexity

In Table 1, we list the time complexities of testing for KNN Offline-1 and KNN Offline-2. Here, n denotes the number of documents to be ranked for the test query, k denotes the number of nearest neighbors, and m denotes the number of queries in the training data [2].

From Table 1, we can see that the time complexities for testing KNN Offline methods mainly lie in the ranking part, which are of order $O(n \log n)$. In this regard, the time complexities of online testing are comparable with that of the single model approach (one that trains a model using all the training data), which is also of order $O(n \log n)$.

As for training, it is clear that the KNN Offline methods have much higher time complexity than the single model approach. Suppose Ranking SVM is used as the learning algorithm. As we know,

[2]Note that we treat k, m and n as variables, and leave other parameters like T as constant in our complexity analysis. Furthermore, the complexities we list in Table 1 are in accordance with our implementations, which are not necessarily the lowest complexities that one could get. Usually m and n can range from thousands to tens of thousands [5]. In our experiments, k can be in the hundreds.

the time complexity of training Ranking SVM is of polynomial order in number of document pairs [10]. We use c to represent the polynomial coefficient. According to [16], $c \approx 1.2$ to 2, depending on the dataset and feature set used. Suppose that p is the average number of document pairs per training query, then the time complexity of training for the single model approach is of order $O((mp)^c)$, while those for the KNN Offline methods are of order $O(m(kp)^c)$. That is to say, the time complexities of training for KNN Offline methods are about $k(k/m)^{(c-1)}$times higher than that of the single model approach. (Note that we can further improve the efficiency of training by running multiple training processes *in parallel*.) However, considering the fact that for a search engine the efficiency requirement on offline processing is usually not as high as that on online processing, we can still say that KNN Offline-1 and KNN Offline-2 are feasible in practice.

3.3.4 Theoretical Analysis

Next, we conduct theoretical analysis to see whether and on which condition KNN Offine-1 and KNN Offline-2 are accurate approximations of KNN Online, on the basis of stability theory. This is very important for running the algorithms in practice.

Definition 1. Uniform leave-one-instance-out stability

Let $\mathcal{X}, \mathcal{Y}$ denote the input and label spaces respectively. $S = \{z_1 = (x_1, y_1), ..., z_p = (x_p, y_p)\} \subset (\mathcal{X} \times \mathcal{Y})^p$ denotes the training set. $S^i = \{z_1, ..., z_{i-1}, z_{i+1}, ..., z_p\} \subset (\mathcal{X} \times \mathcal{Y})^{p-1}$ denotes the training set derived by leaving one instance $z_i (i = 1, ..., p)$ out of S. L is the learning algorithm which can learn a model h_S by minimizing the loss function $\sum_i l(h, z_i)$ on the training set S. Given $\tau : N \rightarrow R$, we say that L has uniform leave-one-instance-out stability τ, if for $\forall z \in \mathcal{X} \times \mathcal{Y}$,

$$|l(h_S, z) - l(h_{S^i}, z)| \leqslant \tau(p). \tag{2}$$

THEOREM 1. *Let S_1, S_2 denote two training sets with p1 and p2 instances respectively, h_{S_1}, h_{S_2} be two models learned from them by using a learning algorithm L. If L has uniform leave-one-instance-out stability τ, then we have $\forall z \in \mathcal{X} \times \mathcal{Y}$,*

$$|l(h_{S_1}, z) - l(h_{S_2}, z)| \leqslant \tau(\min(p_1, p_2))(p_1 + p_2 - 2\triangle), \tag{3}$$

where $\triangle$ is the number of shared instances in S_1 and S_2.

It is easy to verify that Theorem 1 holds, and we omit the proof here.

Theorem 1 states that when the training sets of two models are similar, the models will also be similar in terms of the difference in loss, if the learning algorithm is stable with respect to minor changes in the training examples.

In our case, Ranking SVM is used as the learning algorithm. Accordingly, we have

(1) Ranking SVM is proven to have uniform leave-one-instance-out stability $\tau(p) = \frac{\kappa^2}{\lambda p}$ [1], where λ is a regularization coefficient, $K(,)$ is the kernel used, and $\forall x \in \mathcal{X}, K(x, x) \leq \kappa^2 < \infty$. In this case, the upper bound in Eq.(3) becomes $\frac{\kappa^2}{\lambda}\gamma$, where $\gamma = \frac{(p_1+p_2-2\triangle)}{\min(p_1,p_2)}$.

(2) Suppose that for a test query q, the training set $S_{N_k(q)}$ is used in KNN Online and the training set $S_{N_k(q_{i*})}$ is used in KNN Offline. Thus, we have $p_1 = |S_{N_k(q)}|, p_2 = |S_{N_k(q_{i*})}|$, and $\triangle = |S_{N_k(q_{i*})} \cap S_{N_k(q)}|$. If $S_{N_k(q_{i*})}$ can have a large overlap with $S_{N_k(q)}$ (i.e. γ is very small), then the performances of KNN Online and KNN Offline will be very close to each other, in terms of the difference in their prediction's loss on any document pairs associated with query q. This condition was validated to be true on the datasets in our experiments (refer to Section 4.3.1 for details).

4. EXPERIMENTS

4.1 Experimental Setting

4.1.1 Dataset

In the experiment we used data obtained from a commercial search engine. There are two datasets, one containing 1,500 training queries and 400 test queries (denoted as Dataset 1) and the other containing 3,000 training queries and 800 test queries (denoted as Dataset 2). Each query is associated with its retrieved documents along with labels representing the relevance of those documents. There are five levels of relevance: perfect, excellent, good, fair, and bad. For a query-document pair, a feature vector is defined. There are in total 200 features including term frequency, PageRank, etc. All the methods tested in this section are based on the same feature set.

Note that we did not use the public LETOR data [15], because the numbers of queries in the datasets are too small to conduct meaningful experiments on query dependent ranking.

4.1.2 Parameter Selection

In the experiments, we adopted Ranking SVM [11] as the basic learning algorithm. Ranking SVM has only one parameter λ representing the trade-off between empirical loss and model complexity. We set $\lambda = 0.01$ for all methods.

In KNN, we used BM25 as the reference model to rank documents, chose the top $T = 50$ documents, and then created query features.

As we know, the parameter k in KNN is crucial to the performance of the algorithm. 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, however, we present the testing results with respect to different values of k instead of automatically determining its most appropriate value.

4.1.3 Evaluation Measure

We used NDCG [9] as our evaluation measure. NDCG at position n is calculated as:

$$N(n) \equiv Z_n \sum_{j=1}^{n} \begin{cases} 2^{r(j)} - 1, & j = 1, 2 \\ \frac{2^{r(j)}-1}{\log(j)}, & j > 2 \end{cases} \tag{4}$$

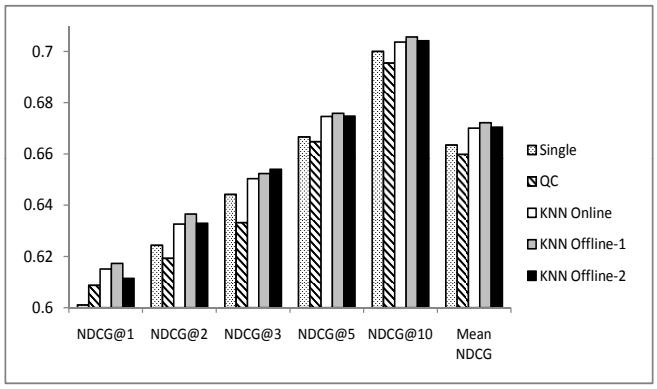
where j is the position in the document list, $r(j)$ is the score of the j-th document in the document list (we represent scores of perfect, excellent, good, fair, and bad as 4, 3, 2, 1, 0, respectively), and Z_n is a normalizing factor. Z_n is chosen so that for the perfect list NDCG at each position equals one.

4.2 Experimental Results

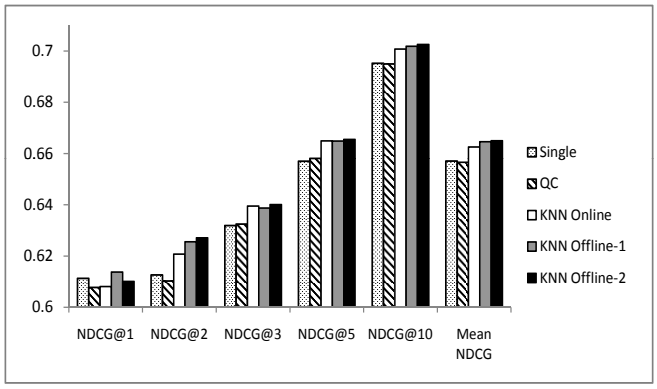
4.2.1 Comparisons with Baselines

We compared the proposed KNN methods with the baselines of the single model approach (denoted as Single) and the query classification based approach (denoted as QC).

For the second baseline, we implemented the query type classifier proposed in [26] to classify queries into three categories (topic distillation, name page finding, and home page finding). Then we trained one ranking model for each category. For a test query, we first applied the classifier to determine its type, and then used the corresponding ranking model to rank its associated documents.

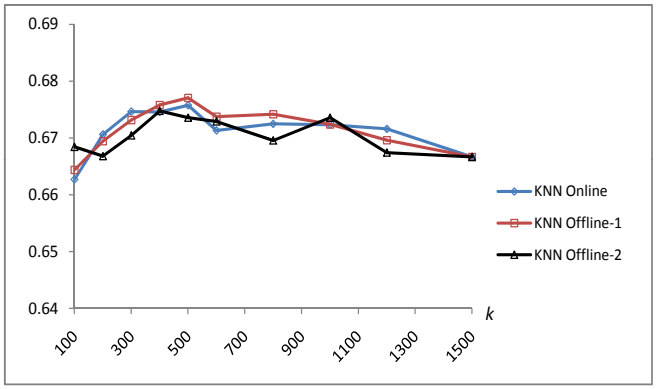


(a) Ranking Accuracies in terms of NDCG (Dataset 1)

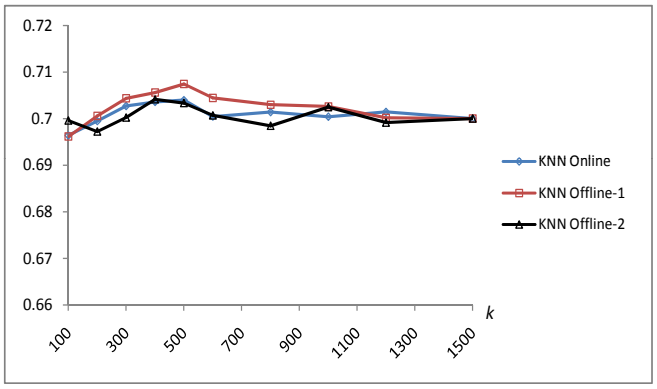


(b) Ranking Accuracies in terms of NDCG (Dataset 2)

Figure 7: Comparisons between KNN Methods and Baselines.



(a) Ranking Accuracies at NDCG@5



(b) Ranking Accuracies at NDCG@10

Figure 8: Performances of KNN Methods with Different k Values on Dataset 1.

In KNN, we set $k = 400$ for Dataset 1, and $k = 800$ for Dataset 2[3]. The experimental results are shown in Figure 7. Note that mean NDCG refers to the mean value of NDCG@1 through NDCG@10 (See also the experiments in [5]).

From Figure 7, we can see that the proposed three methods (KNN Online, KNN Offline-1 and KNN Offline-2) perform comparably well with each other, and all of them almost always outperform the baselines (Single and QC). We conducted t-tests on the improvements in terms of mean NDCG. The results indicate that for both Dataset 1 and Dataset 2, the improvements of the KNN methods over both Single and QC are statistically significant (p-value<0.05). We make the following observations from these results:

(1) The better results of KNN over Single indicate that query dependent ranking does help, and an approach like KNN can indeed effectively accomplish the task.

(2) The superior results of KNN to QC indicate that an approach based on soft classification of queries like KNN is more successful than an approach based on hard classification of queries like QC.

(3) QC cannot work better than Single, mainly due to the relatively low accuracy of query classification. In fact, the accuracies of classification in terms of F1 measure are only about 60% in the two datasets. Errors in the query classification can greatly damage the results of document ranking. This also shows that it is not easy to develop a query dependent ranking method that can beat conventional ranking methods. In contrast, the KNN methods can successfully leverage the ranking patterns of similar queries and achieve better ranking performances.

[3]As discussed in Section 4.2.2, the ranking performance is 'flat' with respect to a large range of k values. Here we simply picked one point from that range and used it in the experiments.

4.2.2 *Effects of Different k Values*

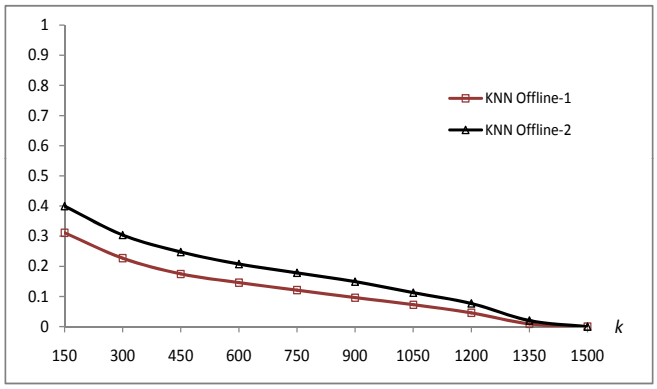
We tested the performances of the KNN methods with different values of parameter k, the number of nearest neighbors selected. Notice that when $k = m$, KNN becomes equivalent to Single, where m denotes the number of training queries.

Figure 8 shows the performances of the proposed methods on Dataset 1 with different k values in terms of NDCG@5 and NDCG@10. From this figure, we can see that as k increases, the performances first increase and then decrease. More specifically,

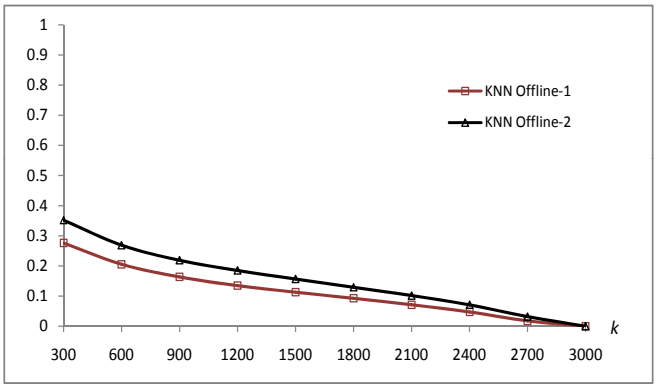
(1) When only a small number of neighbors are used, the performances of KNN are not so good due to the insufficiency of training data.

(2) When the numbers of neighbors increase, the performances gradually improve, because of the use of more information.

(3) However, when too many neighbors are used (approaching 1500, which is equivalent to Single), the performances begin to deteriorate. This seems to indicate that query dependent ranking can really help.



(a) DataSet1



(b) DataSet2

Figure 9: Change Ratio between KNN Online and KNN Offline.

(4) The best performances are achieved when k takes values from a relatively large range, i.e. from 300 to 800 [4].

We obtain similar results on Dataset 2. Due to space limitations, we omit them here.

4.3 Discussion

4.3.1 Validation of Theoretical Analysis

From the theoretical analysis in Section 3.3.4 we know that if the training query sets in KNN Online and KNN Offline have a large overlap (i.e. γ is very small), then the performances of KNN Online and KNN Offline will be very close to each other. We call γ the change ratio.

We tried to verify that this low change ratio assumption holds for the datasets. Since Ranking SVM is used, document pairs become the instances in this case. Figure 9 shows the relationship between k and change ratio on Dataset 1 and Dataset 2. The x-axis denotes k and the y-axis denotes change ratio.

From the figure we can see that the change ratio is always small. Furthermore, when k approaches m, change ratio approaches zero. Given the results,the theoretical discussions in Section 3.3.4 indicate that KNN Online and KNN Offline can have very similar performances. This seems to be verified by the experimental results in Section 4.2 as well (i.e. their test performances are similar on both Dataset 1 and Dataset 2).

[4]We observe the range in our experiments. However, we have no idea whether the same conclusion holds on larger datasets.

Besides, the change ratio of KNN Offline-1 is smaller than KNN Offline-2. This is reasonable since more simplifications have been introduced in KNN Offline-2. Furthermore, we see that the change ratio is smaller on Dataset 2 than on Dataset 1. This seems to be reasonable: as data size increases, the density in the query feature space will become higher, and therefore it will become easier to find similar neighbors. Therefore, the change ratio may be smaller in a larger dataset and the performances of KNN Online and KNN Offline will tend to be more similar.

5. CONCLUSION AND FUTURE WORK

In this paper, we have pointed out that owing to the great variety of queries, ranking of documents in search (particularly web search) should be conducted by using different models based on different properties of queries. We have proposed a K-Nearest Neighbor (KNN) approach to learning ranking functions along this direction. To improve the efficiency of the method we have also devised two methods which conduct training offline. Experimental results show that the proposed approach outperforms both the single model approach and the query classification based approach.

As future work, we plan to investigate the following issues:

(1) We have focused on reducing the complexity of online processing in our method. The complexity of the offline processing is still high. We will investigate the feasibility of conducting local clustering on the training queries (it is also based on the locality property of queries) to further reduce the offline complexity.

(2) The complexity of online processing in the KNN methods can be further reduced if we use KD-trees [19] or other advanced data structures for our nearest neighbor search.

(3) As query features, we utilized one approach. We will try to define other query features and investigate their effects on performance.

(4) We have used Euclidean distance as the metric in our KNN methods. It would be interesting to see whether other metrics can work better for the task. Exploiting metric learning [27] is also a direction we can explore.

(5) We have tried the KNN methods with fixed numbers of neighbors. It is also a common practice to use a fixed radius in KNN. We plan to make an investigation into this as well.

(6) We have studied one approach to query dependent ranking. We plan to examine the many other potentially helpful approaches.

6. REFERENCES

[1] S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *Proceedings of COLT 2005*, pages 32–47, 2005.

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

[3] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and O. Frieder. Varying approaches to topical web query classification. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 783–784, New York, NY, USA, 2007. ACM.

[4] S. M. Beitzel, E. C. Jensen, O. Frieder, D. Grossman, D. D. Lewis, A. Chowdhury, and A. Kolcz. Automatic web query classification using labeled and unlabeled training data. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 581–582, New York, NY, USA, 2005. ACM.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM, 2007.

[7] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[8] D. S. Guru and H. S. Nagendraswamy. Clustering of interval-valued symbolic patterns based on mutual similarity value and the concept of -mutual nearest neighborhood. In *ACCV (2)*, pages 234–243, 2006.

[9] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[10] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2002.

[12] I. Kang and G. Kim. Query type classification for web document retrieval. In *SIGIR '03: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2003.

[13] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, USA, 2001. ACM.

[14] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 391–400, New York, NY, USA, 2005. ACM.

[15] 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 *SIGIR '07: Proceedings of the Learning to Rank workshop in the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[16] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, 7(1):36–43, 2005.

[17] R. Nallapati. Discriminative models for information retrieval. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71, New York, NY, USA, 2004. ACM.

[18] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

[19] F. P. Preparata and M. I. Shamos. *Computational Geometry:Discriminative models An Introduction (Monographs in Computer Science)*. Springer, August 1985.

[20] M. Richardson, A. Prakash, and E. Brill. Beyond PageRank: machine learning for static ranking. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 707–715, New York, NY, USA, 2006. ACM.

[21] S. Robertson. Overview of the okapi projects. In *Journal of Documentation*, pages 275–281, 1998.

[22] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 13–19, New York, NY, USA, 2004. ACM.

[23] G. Salton. *The SMART Retrieval System-Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[24] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *J. ACM*, 15(1):8–36, 1968.

[25] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138, New York, NY, USA, 2006. ACM.

[26] R. Song, J.-R. Wen, S. Shi, G. Xin, T.-Y. Liu, T. Qin, X. Zheng, J. Zhang, G. Xue, and W.-Y. Ma. Microsoft research asia at web track and terabyte track of trec 2004. In *Proceedings of the Thirteenth Text REtrieval Conference Proceedings (TREC-2004)*, 2004.

[27] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in NIPS*, number vol. 15, 2003.

[28] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, New York, NY, USA, 2007. ACM.

[29] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, New York, NY, USA, 2007. ACM.