Ranking with Query-Dependent Loss for Web Search

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

Tie-Yan Liu, Tao Qin
Microsoft Research Asia
{tyliu, taoqin}@microsoft.com

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

ABSTRACT

Queries describe the users' search intent and therefore they play an essential role in the context of ranking for information retrieval and Web search. However, most of existing approaches for ranking do not explicitly take into consideration the fact that queries vary significantly along several dimensions and entail different treatments regarding the ranking models. In this paper, we propose to incorporate query difference into ranking by introducing query-dependent loss functions. In the context of Web search, query difference is usually represented as different query categories; and, queries are usually classified according to search intent such as navigational, informational and transactional queries. Based on the observation that such kind of query categorization has high correlation with the user's different expectation on the result accuracy on different rank positions, we develop position-sensitive query-dependent loss functions exploring such kind of query categorization. Beyond the simple learning method that builds ranking functions with pre-defined query categorization, we further propose a new method that learns both ranking functions and query categorization *simultaneously*. We apply the query-dependent loss functions to two particular ranking algorithms, RankNet and ListMLE. Experimental results demonstrate that query-dependent loss functions can be exploited to significantly improve the accuracy of learned ranking functions. We also show that the ranking function jointly learned with query categorization can achieve better performance than that learned with pre-defined query categorization. Finally, we provide analysis and conduct additional experiments to gain deeper understanding on the advantages of ranking with query-dependent loss functions over other query-dependent ranking approaches and query-independent approaches.

Categories and Subject Descriptors

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

*The work was done when the first author was intern at Microsoft Research Asia

WSDM'10, February 4–6, 2010, New York City, New York, USA.

General Terms

Algorithms, Experimentation, Theory

Keywords

Ranking for Web search, query-dependent loss function, query difference

1. INTRODUCTION

Ranking has become an essential research issue for informational retrieval and Web search, since the quality of a search system is mainly evaluated by the relevance of its ranking results. The task of ranking in the search process can be briefly described as follows. Given a query, the deployed ranking model 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 key problem of search technology is to develop a ranking model that can best represent relevance.

Many models have been proposed for ranking, including the Boolean model [1], vector space model [20], probabilistic model [18] and language model [13, 17]. Recently, there are renewed interests in exploring machine learning methodologies for building ranking models. Many learning-based approaches have been introduced, some popular examples of which include MCRank [15], RankNet [5], RankSVM [11], RankBoost [8], GBRank [26], ListNet [6], ListMLE [23], and IsoRank [27]. These approaches leverage training data, which consists of queries with their associated documents and relevance labels, and machine learning techniques to make the tuning of ranking models theoretically sound and practically effective.

In most of the previous works, the significant difference in queries are not adequately addressed in the context of ranking. This is clearly not appropriate since queries vary largely in semantics [3, 21] and users' search intentions [14, 19]. For example, in a popular taxonomy of Web search, queries can be coarsely categorized as navigational, informational and transactional [4] in terms of search intentions. As another example, queries can vary in terms of relational information needs, including queries for subtopic retrieval [25] and topic distillation [22].

Although query difference is multi-faceted, we observe that query difference usually has tight correlation with the user's different expectation on the result accuracy on different rank positions. Let us elaborate this issue using Broder's "Taxonomy of Web search" [4], which describes query difference based on the search intent of users and classifies queries into three categories: navigational, informational and transactional. In particular, navigational queries are those which

are intended to find a specific Web site that the user has in mind; informational searches are intended to find information about a topic; transactional ones are intended to complete some Web-mediated activities. Therefore, for the navigational and transactional query, the user expects high accuracy on the top one retrieved result; while for the informational query the user looks for more relevant documents among top-K rank positions. This kind of position-sensitive query difference requires respective objectives for the ranking model. Specifically, for the navigational and transactional query, the ranking model should aim to rank the exact Web page that the user is looking for on the top position of the result list; while for the informational query, the ranking model should target at presenting a set of Web pages relevant to the topic of the query on the top-K positions of returned results. The above only illustrates one aspect of the issue, we can similarly consider the issue in the context of subtopic retrieval and topic distillation. In particular, for the subtopic retrieval query, the objective of ranking model should be presenting a set of Web pages covering as many subtopics as possible on the top-K positions of result list; while for topic distillation query, the ranking model should focus on ranking a set of Web pages best representing one single topic among top-K rank positions.

In this paper, we propose to incorporate query difference into ranking by introducing query-dependent loss functions in the learning process. Inspired by the diverse ranking objectives implied by various queries, we apply different loss functions to different queries in learning the ranking function. Since it is difficult and expensive in practice to extract individual objective for each query, we make use of query categorization to represent query difference such that each query category stands for one kind of ranking objective. In this paper, we focus on Broder's taxonomy of Web search [4] and develop position-sensitive query dependent loss functions according to this popular query categorization.

Unfortunately, query categorization may or may not be available at learning time. Accordingly, beyond learning the ranking functions with pre-defined query categories, we develop a new method for learning ranking functions jointly with query categorization without prior knowledge on query categorization. In this new method, the ranking function and query categorization use totally disjointed feature sets.

To evaluate the effectiveness of our proposed approach, we derive the position-sensitive query-dependent loss function based on Broder's taxonomy, and apply it to two popular ranking algorithms, RankNet and ListMLE. Experimental analysis is employed to verify that query-dependent loss function can be exploited to boost the accuracy of ranking for Web search. We also make a comparison on the ranking accuracy between the learning method that trains the ranking function with pre-defined query categorization and that learns the ranking function jointly with query categorization. Moreover, we provide analysis and conduct additional experiments to gain deeper understanding on the advantages of ranking with query-dependent loss functions over other query-dependent or query-independent ranking approaches.

The major contributions of this work include:

- Proposing to incorporate query difference into ranking by introducing query-dependent loss functions.
- Introducing a new methods for learning the ranking function jointly with learning query categorization
- Exploiting the position-sensitive query-dependent loss function on a popular query categorization scheme of Web search and applying it to two specific ranking algorithms, RankNet and ListMLE.

The remaining parts of this paper are organized as follows. Section 2 proposes the general idea of incorporating query difference into ranking by introducing query-dependent loss functions. In section 3, we specifically define a position-sensitive query-dependent loss function based on Broder's query categorization for Web search and apply it to two concrete ranking algorithms. Experiments and discussions are presented in section 4. A comparison between our method and other query dependent methods for ranking is discussed in section 5. We conclude the paper and point out future research directions in section 6.

2. INCORPORATING QUERY DIFFERENCE INTO RANKING

In this section, we propose to incorporate query difference into ranking by introducing query-dependent loss function, and we also outline learning methods for ranking with the query-dependent loss functions.

2.1 Query-Dependent Loss Functions

We formalize the problem of building a ranking model as finding a function $f \in \mathcal{F}$, where $\mathcal{F}$ is a given function class, such that f minimizes the risk of ranking in the form of a given loss function L_f. For a general learning to rank approach, the loss function is defined as:

$$L_f = \sum_{q \in Q} L(f), \tag{1}$$

where Q denotes the set of queries in the training data; $L(f)$ denotes a query-level loss function, which is defined on ranking function f and has the same form among all queries.

Inspired by the diverse ranking objectives implied by the queries, we incorporate query difference into the loss function by applying different loss functions to different queries. This kind of query-dependent loss function is defined as:

$$L_f = \sum_{q \in Q} L(f; q), \tag{2}$$

where $L(f; q)$ is the query-level loss function defined on both query q and ranking function f, and each query has its own form of loss function.

However, it is difficult and expensive in practice to define individual objective for each query. Thus, we take advantage of query categorization to represent query difference, which means each query category stands for one kind of ranking objective. In general, we assume there is a query category space, denoted as $\mathbf{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_m\}$, where $\mathcal{C}_i (i = 1, \cdots, m)$ represents one query category. We also assume a soft query categorization, which means each query can be described as a distribution over this space. We use $P(\mathcal{C}_i|q)$ to denote the probability that query q belongs to the class $\mathcal{C}_i$ with $\sum_{i=1}^{m} P(\mathcal{C}_i|q) = 1$. Thus, the query-dependent loss function of the ranking function f is defined as:

$$L_f = \sum_{q \in Q} L(f; q) \tag{3}$$

$$= \sum_{q \in Q} \left(\sum_{i=1}^{m} P(\mathcal{C}_i|q) L(f; q, \mathcal{C}_i) \right), \tag{4}$$

where $L(f; q, \mathcal{C})$ denotes a category-level loss function defined on query q, ranking function f and q's category $\mathcal{C}$.

2.2 Learning Methods

After constructing the query-dependent loss function by incorporating the information of query categories, we can learn the ranking function by minimize the loss function. The straightforward method is to first obtain the soft categorization for each query, i.e. $P(\mathcal{C}_i|q), i = 1, \cdots, m$, then to learn the ranking function by minimizing the query-dependent loss function (Eq. 4) with known query categorization. In this simple method, the query categorization is obtained independently with learning the ranking function. However, query categorization may not be precise enough and even not be available at the learning time. Thus, we propose a new learning method which considers query categorization as hidden information and optimizes the ranking function jointly with query categorization. Compared to the straightforward method, this method aims to categorize queries for the purpose of minimizing the loss function for ranking. We denote this method as the *unified* method. We will discuss how to specify the learning method with respect to particular query-dependent functions in the following section.

3. APPLYING QUERY-DEPENDENT LOSS TO A SPECIFIC QUERY CATEGORIZATION

In terms of search intentions, queries are usually classified into three major categories according to Broder's taxonomy [4]: navigational, informational, and transactional. Under this taxonomy, a navigational query is intended to locate a specific Web page, which is often the official homepage or subpage of a site; an informational query seeks information on the query topic; and a transactional query seeks to complete a transaction on the Web.

For this taxonomy of Web search, we observe that the rank objectives usually have tight correlation with the user's different expectation on the result accuracy on different rank positions. Specifically, for the navigational and transactional query, the ranking model should aim to retrieve the exact relevant Web page on the top one position in the result sets; while for the informational query, the ranking model should target to rank more relevant Web pages on a set of top positions in the result sets. To incorporate this kind of query difference into ranking, we can define the position-sensitive query-dependent loss function by targeting the ranking objectives to respective sets of ranking positions for different query categories. In particular, for the navigational and transactional query, the loss function should focus on that exact relevant page; while for the informational query, the loss should consider those relevant pages which should be ranked in the range of top-K positions.

In this section, we first define the query-dependent loss functions which represent these position-sensitive objectives. Then, we introduce the learning methods to minimize this position-sensitive query-dependent loss function. Moreover, we present two examples of applying the proposed position-sensitive query-dependent loss function to two concrete ranking algorithms, respectively.

3.1 Query-Dependent Loss Functions Based on Query Taxonomy of Web Search

According to Broder's taxonomy and corresponding ranking objectives discussed above, we classify queries into two categories, i.e., $\mathbf{C} = \{\mathcal{C}_I, \mathcal{C}_N\}$, where $\mathcal{C}_I$ denotes informational queries and $\mathcal{C}_N$ denotes navigational and transactional queries. Note that we combine navigational and transactional queries into $\mathcal{C}_N$ since both of them describe the similar search intention which focuses on the accuracy of top-one ranked result.

According to Eq. 4, the query-dependent loss function is now defined as:

$$L(f; q) = \alpha(q)L(f; q, \mathcal{C}_I) + \beta(q)L(f; q, \mathcal{C}_N), \quad (5)$$

where $\alpha(q) = P(\mathcal{C}_I|q)$ represents the probability that q is an informational query, $\beta(q) = P(\mathcal{C}_N|q)$ indicates the probability that q is a navigational or transactional query, and $\alpha(q) + \beta(q) = 1$.

In general, we can define a position-sensitive loss function $L(f; q, \mathcal{C})$ in the form of:

$$L(f; q, \mathcal{C}) = \sum_{x \in X_q} l(f(x), g(x), p(x); \Phi(q, \mathcal{C})) \quad (6)$$

where X_q is the set of training examples under query q; and x is one of the training examples; the example-level loss l is defined based on ranking function f, ground truth of the example $g(x)$, the true ranking positions of the example $p(x)$ [1], and a set of ranking positions $\Phi(q, \mathcal{C})$ on which users expect high result accuracy. The general principle is that the example-level loss l will contribute to the whole loss if the true rank position $p(x)$ of the example x is included in $\Phi(q, \mathcal{C})$, and the actual value of example-level loss is defined by $f(x)$ and $g(x)$. Specifically, we assume that, the ranking risk of informational query, $L(f; q, \mathcal{C}_I)$, focuses on documents that should be ranked on top-k_I positions, i.e., $\Phi(q, \mathcal{C}_I) = \{1, \cdots, k_I\}$; while the ranking risk of navigational or transactional query targets on documents that should be ranked on top-k_N positions, i.e., $\Phi(q, \mathcal{C}_N) = \{1, \cdots, k_N\}$.

3.1.1 Estimating Rank Positions from Relevance Judgments

As discussed above, in order to build the query-dependent loss function, we need to obtain the true rank position of each training example. The relevance judgments in the training set provide the possibility to obtain the true rank position of each training example. Multi-level relevance judgments are often used in the training set. For example, if all the training examples are labeled using a k-level relevance judgment, the label set contains k distinct relevance levels, such as $\{0, 1, \cdots, k-1\}$, where larger value usually indicates higher relevance.

However, there is an apparent gap between the true rank positions and multi-level relevance judgments. In particular, for some queries, more than one documents may have the same label. In such case, any document x, sharing the same label with some other documents, can be ranked at multiple positions without changing ranking accuracy, i.e. x can have multiple true rank positions $p(x)$. Therefore, it is necessary to find a precise method to map the relevance labels into rank positions. A general method is to utilize labels to estimate the probability that one document is ranked at each position under the given query, such that all the documents with the same label have equal probability to be ranked at the same position, and those with better relevance labels have higher probability to be ranked at higher positions.

There are many specific ways for implementing this general method. In this paper, we introduce one based on the *equivalent correct permutation set*. Given a query q,

[1]True rank position $p(x)$ is the position of example x if we rank all examples under the same query by their ground truth.

assume all the documents under q are labeled using a k-level relevance judgment; and for each label level $t, (t \in \{0, 1, \cdots, k-1\})$, assume there are n_t documents under q whose labels are t. For the document list under q, we define an *equivalent correct permutation set* S:

$$S = \{\tau | g(x_i) > g(x_j) \Rightarrow \tau(x_i) < \tau(x_j)\}, \quad (7)$$

which means, for each permutation $\tau \in S$, if the relevance label of one document x_i is better than another document x_j, i.e. $g(x_i) > g(x_j)$, then the position of x_i in τ is higher than that of x_j, i.e., $\tau(x_i) < \tau(x_j)$. Then, the probability that a document x with label t is ranked at certain position ρ can be defined as:

$$P(p(x) = \rho) = \frac{1}{|S|} \sum_{\tau \in S} \mathbf{1}_{\{p(x)=\rho \text{ in } \tau\}}, \quad (8)$$

where $\mathbf{1}_{\{p(d)=\rho \text{ in } \tau\}}$ is an indicator function which equals 1 if document x is at position ρ in permutation τ and otherwise 0. Then, the probability can be calculated as:

$$P(p(x) = \rho) = \begin{cases} \frac{1}{n_t} & , \quad 1 + \sum_{m=t+1}^{k-1} n_m \leq \rho \leq \sum_{m=t}^{k-1} n_m \\ 0 & , \quad \text{otherwise} \end{cases}$$

For example, assume under a query q, there are five documents $\{a, b, c, d, e\}$. A three-level labeling is used. Assume the label set is $\{0, 1, 2\}$ where 2 means highest relevance. Assume the labels of five documents are $\{2, 2, 1, 1, 0\}$ respectively. Based on the above method, both a and b have probability 50% to be ranked at position 1 and 2, and 0 at other positions; both c and b have probability 50% to be ranked at position 3 and 4, and 0 at other positions; e has probability 100% to be ranked at position 5.

3.2 Learning Methods

To learn the ranking function, we need to minimize the query-dependent loss function with respect to the ranking parameters, denoted as ω. We use f_ω to represent the ranking function defined by the parameter vector ω. To this end, the query dependent loss function can be defined as:

$$L_f = \sum_{q \in \mathcal{Q}} \alpha(q) L(f_\omega; q, \mathcal{C}_I) + \beta(q) L(f_\omega; q, \mathcal{C}_N) \quad (9)$$

The straightforward method is to first obtain pre-defined categorization for each query and then learn the parameters of ranking function with pre-defined query categorization. In particular, for pre-defined informational queries, $\alpha(q) = 1, \beta(q) = 0$; while for pre-defined navigational or transactional queries, $\alpha(q) = 0, \beta(q) = 1$. If there exists soft query categorization, i.e. $\alpha(q)+\beta(q) = 1$ and $\alpha(q), \beta(q) > 0$, we can also use them directly in the loss function.

According to Eq. 6, the position-sensitive query-dependent loss function can be represented as the sum of example-level loss. We assume the original loss function are convex. Since there is no new parameters for pre-defined categorization, and all the example-level loss are part of original loss function, the proposed query-dependent loss function are convex as well. Therefore, we can apply the gradient descent method with respect to parameters of the ranking function to minimize the query-dependent loss function.

• ***Unified* Method:**

Due to the fact that existing query categorization may not be best for ranking and that even this kind of knowledge may not be available at learning time, we propose a new method learning the ranking function jointly with query categorization. We refer to this new method as **unified** method.

Algorithm 1: **Unified Learning Method**

input : a set of training examples for learning to rank; queries defined by query features.

output: parameter vector of the ranking function, ω, and parameter vector of the query categorization, γ, which minimize the loss function L_f.

Algorithm:

Start with an initial guess, e.g. random values, for ω and γ;

begin

while $(L_f(\omega_k, \gamma_k) - L_f(\omega_{k+1}, \gamma_{k+1})) > \epsilon$ **do**

• ***Step 1: Learning*** ω

using gradient descent to minimize L_f with respect to ranking function parameters ω, holding the query categorization parameters fixed, i.e.,

$$\omega_{k+1} \leftarrow \arg\min_{\omega} \sum_{q \in \mathcal{Q}} \alpha_{\gamma_k}(q) L(f_{\omega_k}; q, \mathcal{C}_I) + \beta_{\gamma_k}(q) L(f_{\omega_k}; q, \mathcal{C}_N)$$

• ***Step 2: Learning*** γ

using gradient descent to minimize L_f with respect to query categorization parameters γ, holding the ranking function parameters fixed, i.e.,

$$\gamma_{k+1} \leftarrow \arg\min_{\gamma} \sum_{q \in \mathcal{Q}} \alpha_{\gamma_k}(q) L(f_{\omega_{k+1}}; q, \mathcal{C}_I) + \beta_{\gamma_k}(q) L(f_{\omega_{k+1}}; q, \mathcal{C}_N)$$

end

Considering that query categorization, as hidden information in learning, is defined by a set of query features, which are disjointed with features of the ranking function. We assume $\mathbf{z}_q$ is the feature vector of query q and γ is the vector of parameters of query categorization, and we use the logistic function to obtain the query categorization $\alpha_\gamma(q)$ and $\beta_\gamma(q)$ from query features:

$$\alpha_\gamma(q) = \frac{\exp(\langle \gamma, \mathbf{z}_q \rangle)}{1 + \exp(\langle \gamma, \mathbf{z}_q \rangle)}, \quad \beta_\gamma(q) = \frac{1}{1 + \exp(\langle \gamma, \mathbf{z}_q \rangle)}, \quad (10)$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner product.

Therefore, the query-dependent loss function can be represented as:

$$L_f = \sum_{q \in \mathcal{Q}} \alpha_\gamma(q) L(f_\omega; q, \mathcal{C}_I) + \beta_\gamma(q) L(f_\omega; q, \mathcal{C}_N) \quad (11)$$

which contains both ranking parameters ω and query categorization parameters γ. We need to minimize with respect to both ω and γ. We introduce a new algorithm, as shown in Algorithm. 1, to alternates between minimizing the loss with respect to ω and γ. Similar to the straightforward learning method, we use gradient descent methods in each iteration.

Remark: Note that, since we do not need information of query categories during testing, γ will not be used for ranking during testing; but γ can be used to compute the query categorization of testing queries.

Then, let's look at an algorithmic property of the unified learning method before applying it to specific ranking models.

Theorem 1. *Algorithm 1 converges in a finite number of steps if the original loss function, from which the query-dependent loss function is derived, is convex and bounded below.*

PROOF. As discussed above, the query-dependent loss function is derived from an original loss function $L_0(f;q)$ of one particular ranking model. We assume the original loss function $L_0(f;q)$ is convex and bounded below, i.e., $L_0(f;q) \geq c > -\infty$.

To prove the convergence of Algorithm 1, it is necessary to show that the algorithm can decrease the loss in each iteration and that the loss is bounded. First, in the step of *Learning* ω, according to Eq. 6, $L(f_\omega;q,\mathcal{C})$ has the same form of example-level loss with L_0 but assigns different predefined weights for different examples according to query categorization. Thus, $L(f_\omega;q,\mathcal{C})$ is convex with respect to ω due to L_0 is convex. To this end, we can use gradient descent to minimize the loss function with respect to ω based on fixed γ. Similarly, in the step of *Learning* γ, since $\alpha_\gamma(q)$ and $\beta_\gamma(q)$ in Eq. 10 are convex with respect to γ, we can use gradient descent to minimize the loss with respect to γ based on fixed ω. Therefore, our algorithm can decrease the loss function in each iteration.

Then, due to the same form of example-level loss with L_0, $L(f_\omega;q,\mathcal{C})$ has the lower bound since L_0 is bounded. And, it is easy to verify that $\alpha_\gamma(q)$ and $\beta_\gamma(q)$ are bounded because $\alpha_\gamma(q),\beta_\gamma(q) > 0$ and $\alpha_\gamma(q)+\beta_\gamma(q)=1$. Given that $\alpha_\gamma(q)$, $\beta_\gamma(q)$ and $L(f;q,\mathcal{C})$ are all bounded, it can be verified that the query-dependent loss function has the lower bound.

In a conclusion, given the fact that our algorithm can decrease the query-dependent loss function in each iteration and that the query-dependent loss function has the lower bound, we can prove that Algorithm 1 can converge in a finite number of steps. Suppose that under the initial guess of ω and γ, the loss function has the value of $\mathcal{L}$, since the decreasing of the loss function is more than ϵ in each iteration except the last one, the algorithm can converge using no more than $\frac{\mathcal{L}}{\epsilon}$ steps. □

Currently, for many of existing popular ranking algorithms, such as RankNet [5], RankSVM [11], RankBoost [8], ListNet [6], and ListMLE [23], their original loss functions are convex and bounded below. Therefore, we can apply our proposed query-dependent loss function to these ranking algorithms. In the following of this section, we will apply the position-sensitive query-dependent loss function to two particular ranking algorithms, RankNet [5] and ListMLE [23].

3.3 Example Query-Dependent Loss Functions

3.3.1 Example I: Query-Dependent Loss Functions for RankNet

RankNet [5] uses a loss function that depends on the difference of the outputs of pairs of training samples $x_i \succ x_j$ which indicates x_i should be ranked higher than x_j. The loss function is minimized when the document x_i with a higher relevance label receives a higher score, i.e., when $f(x_i) > f(x_j)$.

Let P_{ij} denote the probability $P(x_i \succ x_j)$, and let $\bar{P}_{ij}$ denote the desired target values. Define $o_i \equiv f(x_i)$ and $o_{ij} \equiv f(x_i) - f(x_j)$. RankNet uses the cross entropy loss function [5]:

$$L(o_{ij}) = -\bar{P}_{ij}\log P_{ij} - (1-\bar{P}_{ij})\log(1-P_{ij}),$$

where the map from outputs to probabilities are modeled using a logistic function [2]: $P_{ij} \equiv e^{o_{ij}}/(1+e^{o_{ij}})$. Then the final cost becomes: $L(o_{ij}) = -\bar{P}_{ij}o_{ij} + \log(1+e^{o_{ij}})$.

For a pair of documents, $\langle x_i, x_j\rangle$, assume $p(i)$ and $p(j)$ are true ranking positions for x_i and x_j, respectively; $L(o_{ij})$ will be added into the loss function for navigational and transactional queries if only $p(i) \in \Phi(q,\mathcal{C}_N)$ or $p(j) \in \Phi(q,\mathcal{C}_N)$; Similarly, $L(o_{ij})$ will be added into the total loss for informational queries if only $p(i) \in \Phi(q,\mathcal{C}_I)$ or $p(j) \in \Phi(q,\mathcal{C}_I)$. To this end, the position-sensitive query-dependent loss function of RankNet for each pair can be defined as:

$$\begin{aligned} L(o_{ij},q) &= \textstyle\sum_{p(i)=1}^{n_q} P(p(i)|x_i,g(x_i))(\alpha(q)\cdot \mathbf{1}_{\{p(i)\in\Phi(q,\mathcal{C}_I)\}} \\ &\quad +\beta(q)\cdot \mathbf{1}_{\{p(i)\in\Phi(q,\mathcal{C}_N)\}})\cdot L(o_{ij}), \end{aligned}$$

where n_q is the number of associated documents for query q; $P(p(i)|x_i,g(x_i))$ is the probability that x_i with label $g(x_i)$ is ranked at position $p(i)$, which is calculated using the method in Section 3.1.1.

3.3.2 Example II: Query-Dependent Loss Functions for ListMLE

ListMLE [23] learns a ranking function by taking individual lists as instances and minimizing a loss function defined on the predicted list and the truth list. In particular, ListMLE formalizes learning to rank as a problem of minimizing the likelihood function of a probability model. ListMLE seeks to minimize top-k surrogate likelihood loss [23], which is defined as:

$$L(f;q) = \phi(\Pi_f(\mathbf{x}),\mathbf{y}) = -\log P_{\mathbf{y}}^k(\Pi_f(\mathbf{x}))$$

where $\mathbf{x} = \{x_1,\cdots,x_n\}$ is the list of documents, and n is the number of associated document for query q; $\mathbf{y} = \{y(1),\cdots,y(n)\}$ is the true permutation of documents under q, and $y(i)$ denotes the index of document which is ranked at position i; ϕ is a surrogate loss function; $\Pi_f(\mathbf{x}) = \{f(x_1),\cdots,f(x_n)\}$ denotes the permutation ordered by ranking function f; and $P_{\mathbf{y}}^k(\Pi_f(\mathbf{x}))$ is defined as:

$$P_{\mathbf{y}}^k(\Pi_f(\mathbf{x})) = \prod_{j=1}^{k}\frac{\exp(f(x_{y(j)}))}{\sum_{t=j}^{n}\exp(f(x_{y(t)}))}$$

where k is the parameter which infers that parameterized negative top-k log-likelihood with Plackeet-Luce model is used as the surrogate loss [6].

To build the position-sensitive query-dependent loss function, top-k_N surrogate likelihood loss is used for navigational or transactional queries, i.e. $\Phi(q,\mathcal{C}_N) = \{1,\cdots,k_N\}$; while top-$k_I$ surrogate likelihood loss is used for informational queries, i.e. $\Phi(q,\mathcal{C}_I) = \{1,\cdots,k_I\}$. To this end, the query-dependent loss function of ListMLE for each query can be defined as:

$$L(f;q) = -\alpha_q \log P_{\mathbf{y}}^{k_I}(\Pi_f(\mathbf{x})) - \beta_q \log P_{\mathbf{y}}^{k_N}(\Pi_f(\mathbf{x})) \quad (12)$$

Note that ListMLE has integrated rank positions into its loss function, we do not need to additionally estimate rank positions from relevance labels.

To learn the ranking functions using query-dependent loss functions for RankNet and ListMLE, we employ both the straightforward method with pre-defined query categorization and our proposed *unified* learning methods. In the straightforward method, we use the gradient descent method, which has been used to learn original RankNet [5] and ListMLE [23], to minimize the query-dependent loss function. In our proposed *unified* learning methods, we apply Algorithm 1

to both RankNet and ListMLE. The computation details of this method on RankNet and ListMLE, though tedious, are rather standard and will not be presented here.

4. EXPERIMENTS

4.1 Experimental Setup

In our experimental study, we use the publicly available LETOR 3.0 data set [16] which is a benchmark data set for research on learning to Rank. We use TREC2003 and TREC2004 in LETOR 3.0 to evaluate the performance of learning to rank using query-dependent loss functions. Both of these two tracks categorize all the queries into three search tasks: topic distillation, homepage finding and named page finding. According to their characteristics, we view topic distillation queries as informational queries while homepage finding and named page finding queries as navigational or transactional queries. The statistics of the queries, feature definitions and relevance judgments for the Letor data set can be found in [16].

To define the features of queries (i.e., the $\mathbf{z}_q$ vector used in Eq. 10), we simply follow the heuristic method proposed in [9]. For each query q, we use a reference model (BM25 in this paper) to find its top-T ranked documents, and take the mean of the feature values of the T documents as the features of the query. In our work, we set $T = 50$.

To evaluate the performance of query-dependent loss functions for learning to rank, we compare the ranking methods as shown below.

- **RankNet:** In this method, we apply RankNet [5], which uses original query-independent loss function for ranking. This method has been showed in [5] to have good performance in ranking.
- **SQD-RankNet:** In this method, we apply query-dependent loss function for RankNet, with pre-defined query categorization. And, we adapt the straightforward learning method, as mentioned in section 3.2, to learn the ranking function. We denote this method as a *simple* query-dependent (SQD) method.
- **UQD-RankNet:** In this method, we apply query-dependent loss function for RankNet, without query categorization. And, we adapt the *unified* learning method, as proposed in Alg 1, to learn the ranking function as well as query categorization simultaneously.
- **ListMLE:** In this method, we apply ListMLE [23], which uses original query-independent loss function for ranking. This method has been shown in [23] to have better performance even than many state-of-the-art ranking methods.
- **SQD-ListMLE:** In this method, we apply query-dependent loss function for ListMLE, with pre-defined query categorization. And, we adapt the straightforward method, as mentioned in section 3.2, to learn the ranking function. We also denote it as a *simple* query-dependent (SQD) method.
- **UQD-ListMLE:** In this method, we apply query-dependent loss function for ListMLE, without query categorization. And, we adapt the *unified* method, as proposed in Alg 1, to learn the ranking function as well as query categorization simultaneously.

In the experiments, we adapt two IR metrics, Normalized Discounted Cumulative Gain (NDCG) [10] and Mean Average Precision (MAP), to evaluate the performance of the learned ranking functions.

For a ranked list of documents, the NDCG score [10] at position n is calculated as follows:

$$\text{NDCG}(n) \equiv Z_n \sum_{j=1}^{n} \begin{cases} 2^{r(j)} - 1 & , \quad j = 1 \\ \frac{2^{r(j)}-1}{\log(j)} & , \quad j > 1 \end{cases} \tag{13}$$

where j is the position in the document list, $r(j)$ is the rating of the j-th document in the list (we represent the rating of *relevant* and *irrelevant* as 1 and 0 respectively in this paper), and Z_n is the normalization factor which is chosen so that the perfect list gets a NDCG score of 1.

MAP is calculated as:

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

where Qr 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.

4.2 Experimental Results

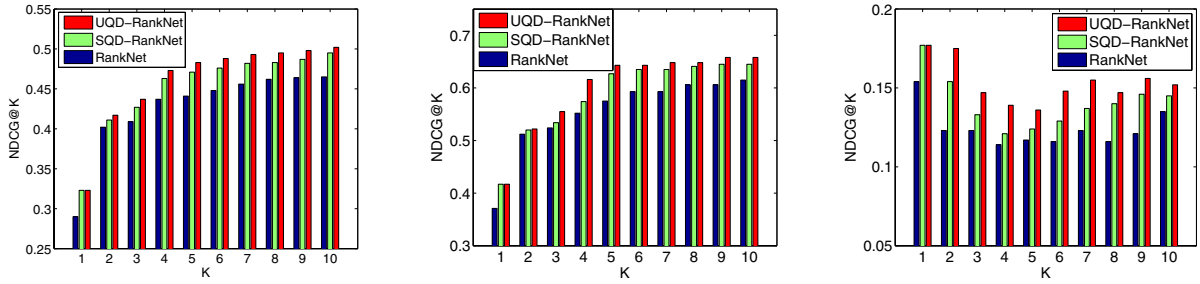
We evaluate the performance of the ranking methods on TREC2003 and TREC2004 in LETOR. For each of these two data set, we conduct 5-fold cross-validation experiments. To ensure both navigational and informational queries have the same distribution in the 5 folds, we split navigational and informational queries into 5 folds, respectively. In SQD-RankNet, UQD-RankNet, SQD-ListMLE, and UQD-ListMLE, we set $k_N = 1$ and $k_I = 10$, i.e., $\Phi(q, \mathcal{C}_N) = \{1\}$ and $\Phi(q, \mathcal{C}_I) = \{1, \cdots, 10\}$.

4.2.1 Performance of query-dependent RankNet

Figure 1 and 2 illustrate the NDCG values of both UQD-RankNet and SQD-RankNet compared with RankNet on TREC2003 and TREC2004, respectively. From Figure 1(a) and 2(a), we observe that these two query-dependent RankNet methods outperform the original RankNet on both data sets, and UQD-RankNet achieves better performance than SQD-RankNet. We also conduct t-test on the improvements in terms of mean NDCG, and the results indicate that for both data sets, the improvements of UQD-RankNet and SQD-RankNet over RankNet are statistically significant (p-value$<$ 0.04).

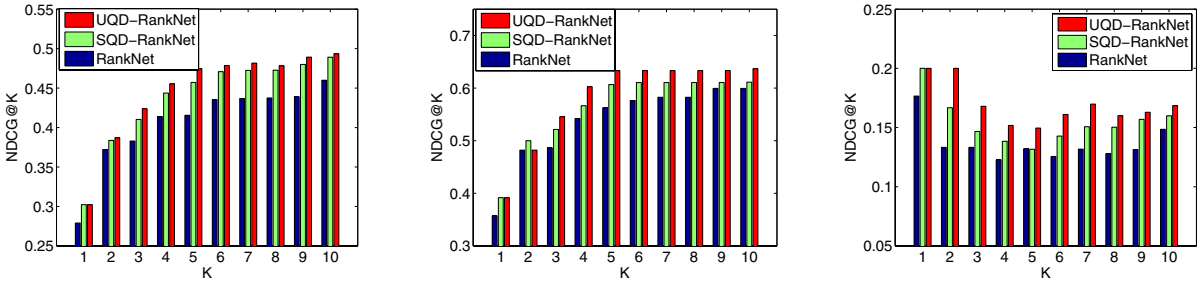
We also test the performance of ranking methods on the respective query types. From Figure 1(b), 1(c), 2(b), and 2(c), we observe that, on both TREC2003 and TREC2004 data sets, UQD-RankNet and SQD-RankNet give higher value on NDCG than RankNet for navigational and informational queries. In particular, for navigational queries, UQD-RankNet and SQD-RankNet perform better especially on NDCG@1 than that exhibited by RankNet; and for informational queries, NDCG@1 $\sim$ 10 gained by UQD-RankNet and SQD-RankNet are much higher than those exhibited by original RankNet. Moreover, UQD-RankNet outperforms SQD-RankNet for both navigational and informational queries.

In Table 1, we demonstrate the MAP value of UQD-RankNet and SQD-RankNet compared with RankNet on TREC2003 and TREC2004, respectively. From these two tables, we can see that query-dependent RankNet methods perform better than original RankNet, and UQD-RankNet reaches much better performance than SQD-RankNet. And,



(a) All query types (b) Navigational queries (c) Informational queries

Figure 1: Accuracy in terms of NDCG of query-dependent RankNet compared with original RankNet on TREC2003



(a) All query types (b) Navigational queries (c) Informational queries

Figure 2: Accuracy in terms of NDCG of query-dependent RankNet compared with original RankNet on TREC2004

Table 1: MAP value of RankNet, SQD-RankNet, and UQD-RankNet

TREC2003			
	All Queries	Navigational	Informational
RankNet	0.372	0.501	0.135
SQD-RankNet	0.381	0.518	0.138
UQD-RankNet	**0.386**	**0.525**	**0.143**
TREC2004			
	All Queries	Navigational	Informational
RankNet	0.365	0.495	0.139
SQD-RankNet	0.375	0.511	0.142
UQD-RankNet	**0.381**	**0.517**	**0.149**

Table 2: MAP value of ListMLE, SQD-ListMLE, and UQD-ListMLE

TREC2003			
	All Queries	Navigational	Informational
ListMLE	0.398	0.512	0.182
SQD-ListMLE	0.410	0.528	0.198
UQD-ListMLE	**0.418**	**0.537**	**0.210**
TREC2004			
	All Queries	Navigational	Informational
ListMLE	0.388	0.509	0.292
SQD-ListMLE	0.402	0.522	0.318
UQD-ListMLE	**0.410**	**0.530**	**0.334**

the t-test result shows that the improvement are statistically significant (p-value < 0.05).

4.2.2 *Performance of query-dependent ListMLE*

Figure 3 and 4 demonstrate the NDCG values for both UQD-ListMLE and SQD-ListMLE compared with ListMLE on TREC2003 and TREC2004, respectively. From Figure 3(a) and 4(a), we observe that these two query-dependent ListMLE methods outperform the original ListMLE on both data sets, furthermore, UQD-ListMLE gives better performance than SQD-ListMLE. After conducting t-test in terms of mean NDCG, we find that, for both data sets, the improvements of UQD-ListMLE and SQD-ListMLE over ListMLE are statistically significant (p-value< 0.04).

We also test the performance of ranking methods on respective query types. From the Figure 3(b), 3(c), 4(b), and 4(c), we can see that, on both TREC2003 and TREC2004 dataset, UQD-ListMLE and SQD-ListMLE give higher value on NDCG than ListMLE for navigational and informational queries. In particular, for navigational queries, UQD-ListMLE and SQD-ListMLE performance better especially on NDCG@1 than that exhibited by ListMLE; and for informational queries, NDCG@1 $\sim$ 10 gained by UQD-ListMLE and SQD-ListMLE are much higher than those exhibited by original RankNet. Moreover, UQD-ListMLE outperforms SQD-ListMLE for both navigational and informational queries.

In Table 2, we illustrate the MAP value of UQD-ListMLE and SQD-ListMLE compared with ListMLE on TREC2003 and TREC2004, respectively. From these two tables, we can see that query-dependent ListMLE methods perform better than original ListMLE, and UQD-ListMLE reaches much better performance than SQD-ListMLE. And, we conduct the t-test whose result prove that the improvement are statistically significant (p-value < 0.05).

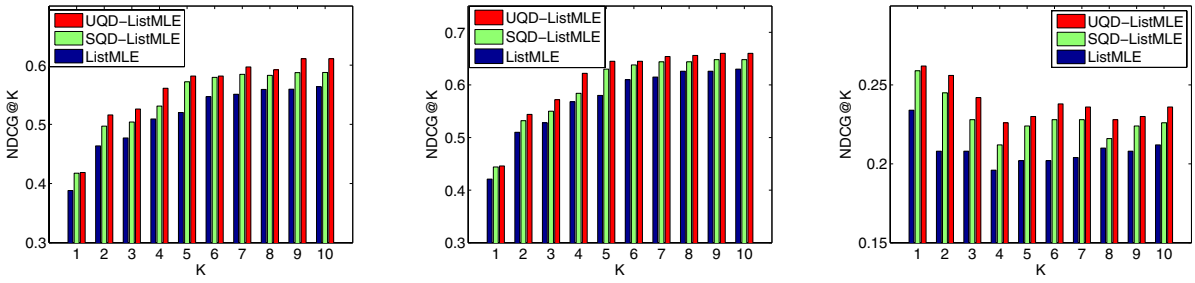
From these results, we can also find that ListMLE and query-dependent ListMLE outperform RankNet and query-dependent RankNet, respectively, which illustrates that listwise ranking models can achieve better performance than pairwise ranking models.

4.2.3 *Effects of Different Parameter Settings*

In this experiment, we explore the effects of different settings of parameters k_N and k_I for ranking and perform comparison study by varying the value of k_N or k_I. We will illustrate the results performed on RankNet, and the experiment on ListMLE shows the similar results.

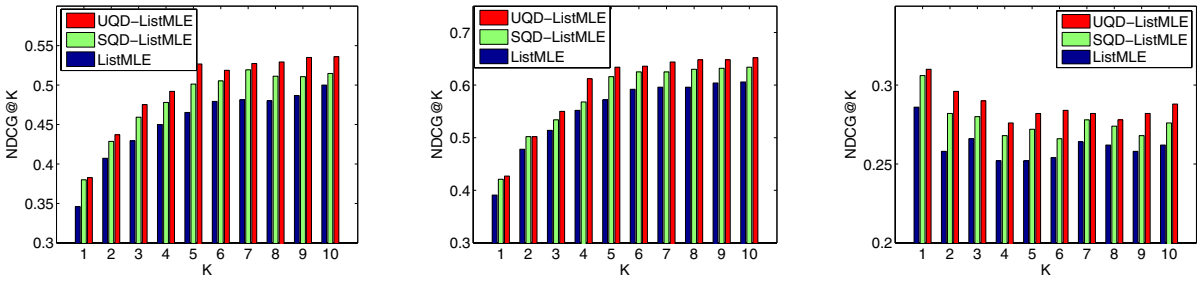
Figure 5 illustrates the performance of UQD-RankNet and SQD-RankNet on navigational queries of TREC2004 against varying the value of k_N. From the figure, we can find that increasing k_N can result in the decreasing performance of ranking for navigational queries.

Figure 6 demonstrates the performance of UQD-RankNet on informational queries on TREC2004 against varying the value of k_I. We can observe that, the accuracy of top-one position, in terms of NDCG@1, remains stable for varying



(a) All query types (b) Navigational queries (c) Informational queries

Figure 3: Accuracy in terms of NDCG of query-dependent ListMLE compared with original ListMLE on TREC2003



(a) All query types (b) Navigational queries (c) Informational queries

Figure 4: Accuracy in terms of NDCG of query-dependent ListMLE compared with original ListMLE on TREC2004

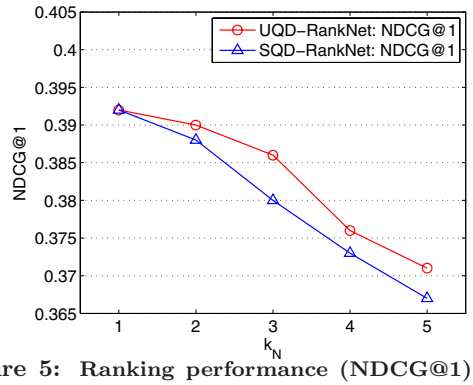


Figure 5: Ranking performance (NDCG@1) of UQD-RankNet and SQD-RankNet on navigational queries of TREC2004 against varying k_N

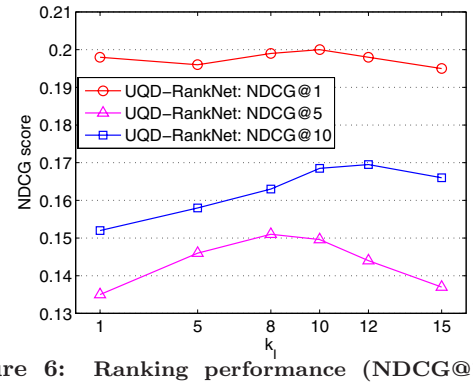


Figure 6: Ranking performance (NDCG@1,5,10) of UQD-RankNet on informational queries of TREC2004 against varying k_I

k_I; too big or too small k_I can both cause the decreasing accuracy of other rank positions for informational queries.

4.3 Discussions

4.3.1 *Availability of Query-Specific Information in Training and Testing*

As presented in Section 2.2, we use explicit query categories (straightforward learning method) or query-specific features (*unified* learning method) to learn the ranking models with query-dependent loss functions. However, when we employ the learned ranking model to perform ranking on new queries, we do not use any information of query classes or query-specific features for the new query. We hypothesize the reason that ranking models using query-dependent loss functions can outperform the original ranking models even without using query-specific information at *query time* as follows: Although query-specific classes and features are not available at query time, they can be viewed as extra tasks for the learner. Therefore, these query-specific information of training data set are transferred into other common features as training signals. We can benefit from ranking models with query-dependent loss functions due to the information in the extra training signals serving as a query-specific inductive bias for ranking [7].

4.3.2 *Query Categorization in Unified Learning Methods*

The above experimental results illustrate that it is possible to achieve better ranking performance by using the *unified* learning method than the straightforward method. We give two possible explanations. First, the straightforward method uses hard query categorization in learning, however, some queries may not be purely navigational or informational; therefore, the *unified* method using soft query categorization can define the more precise loss function for each query. Second, the pre-define query categorization may not be best for ranking. The *unified* method learns query categorization in tandem with the optimization of the ranking function, such that the obtained query categorization, even if contradicted with pre-defined categorization, can give more contribution to building precise ranking models than the straightforward method. In the following, we will demonstrate two specific examples on these two explanations.

Table 3: Top 10 Results (Doc IDs) of One Informational Query (Query ID: 87)

RankNet	SQD-RankNet	UQD-RankNet
16116	61542	61542
151538	151538	**183864**
139846	156752	16116
61542	139846	156986
183864	**183864**	**165686**
156752	141943	**397632**
141943	**165686**	462230
33723	33723	**156335**
158984	158984	151538
165686	**156335**	144691

Table 4: Top 10 Results (Doc IDs) of One Navigational Query (Query ID: 52)

RankNet	SQD-RankNet	UQD-RankNet
2003	13701	**2003**
13701	**2003**	**93720**
93720	124621	13701
542787	220784	**68046**
178575	14266	**60571**
39599	**93720**	124621
68046	149052	220784
124621	59495	8889
8889	**68046**	14266
60571	542787	149052

Table 3 and 4 show the top ten results (Doc IDs) of RankNet, SQD-RankNet and UQD-RankNet for the query with ID 87 and those with ID 52, respectively. The Doc IDs of the relevant documents for these two queries are bold and red colored.

Query 87 is categorized as informational in TREC2004. Thus, it is 100% of informational and 0% of navigational in SQD-RankNet. However, learning using UQD-RankNet renders this query about 63% informational and about 37% navigational. From the table 3, we can find that UQD-RankNet outperforms SQD-RankNet and RankNet for this query. In particular, UQD-RankNet and SQD-RankNet boost more relevant results into the top 10 rank positions by using query-dependent loss functions; Furthermore, UQD-RankNet boost relevant results to higher rank positions than SQD-RankNet. It illustrates that we can boost the accuracy of ranking by incorporating soft query categorization in learning to rank. Note that such soft categorization is learned jointly with learning ranking function.

Query 52 is categorized as navigational in TREC2004. Thus, it is 0% of informational and 100% of navigational in SQD-RankNet. However, after learning using UQD-RankNet, this query is about 58% to be informational and about 42% to be navigational, which means UQD-RankNet consider this query more like an informational query. From the table 4, we can find that UQD-RankNet outperforms SQD-RankNet and RankNet for this query. Moreover, even RankNet outperforms SQD-RankNet for this query, which indicates that it may decrease the accuracy of ranking by considering this query as totally navigational. This example demonstrates that what is good in terms of query categorization is not necessarily good in terms of ranking, and we can boost the accuracy of ranking by using query categorization learned jointly with learning ranking function.

5. QUERY-DEPENDENT LOSS FUNCTIONS V.S. QUERY-DEPENDENT RANK FUNCTIONS

The importance of query-dependent ranking is now widely recognized. Several previous studies have exploited another query-dependent method: query-dependent ranking functions. The key idea of those efforts is to employ different ranking functions for different queries. Kang et al. [12] classified queries into two categories based on search intentions and built two different ranking models accordingly. Geng et al. [9] proposed a K-Nearest Neighbor (KNN) method to employ different ranking models for different queries. In addition, Zha et al. [24] tried another query-dependent method, which implicitly incorporates query differences using monotone transformations of the learned ranking functions.

In this section, we compare the performance between the ranking function using query-dependent loss function and query-dependent ranking function. We employ RankNet to construct individual ranking functions for navigational queries and informational queries, respectively. We denote this approach as QC-RankNet.

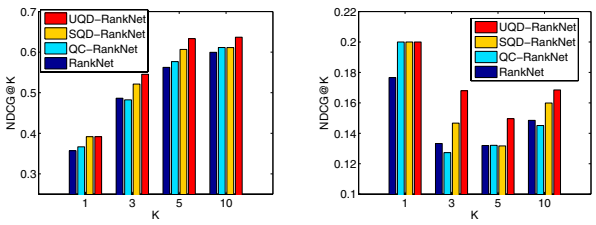
Figure 7 illustrates the performance of UQD-RankNet, SQD-RankNet, QC-RankNet and RankNet on navigational and informational queries of TREC2004, respectively. Notice that QC-RankNet for navigational queries is trained using only the navigational queries with associated documents from the training set, and so is QC-RankNet for the informational queries; While the other three methods are learned using the whole training set. From these figures, we can find that ranking functions using query-dependent loss function (UQD- and SQD-RankNet) outperform query-dependent ranking function (QC-RankNet) and query-independent ranking (RankNet). After conducting t-test on the improvements in terms of mean NDCG, we find that the improvements of UQD-RankNet and SQD-RankNet over QC-RankNet are statistically significant (p-value<0.05).

Why ranking functions learned using query-dependent loss functions can achieve better performance than the corresponding query-dependent ranking functions as well as query-independent ranking functions? We hypothesize a couple of reasons as follows.

Firstly, query-dependent loss function contains more useful information for ranking than the loss for query-independent ranking function or query-dependent ranking function. For query-independent approaches, we use the same loss function for all the query categories, i.e., we minimize $L(Q_I) + L(Q_N)$, where Q_I and Q_N represent the training group of informational and navigational queries, respectively; for query-dependent ranking functions, we use respective loss functions for each query category, i.e., we minimize $L_I(Q_I)$ and $L_N(Q_N)$ separately; however, for our proposed approach, we combine both the loss function for informational queries and that for navigational queries together on each query, but they are weighted according to the query's soft categorization, i.e., we minimize $\sum_{q \in Q_I \bigcup Q_N} \alpha(q)L_I(q) + \beta(q)L_N(q)$. Thus, we incorporate more information into query-dependent loss functions than the other two approaches.

Secondly, due to many queries that can fit into more than one categories, the ranking function using query-dependent loss functions outperform query-dependent ranking functions. For example, with the query "*united nations*", the user may expect to get the homepage of the United Nations (a navigational intention), or the user may be looking for recent news regarding a United Nations resolution (an informational intention). For this kind of queries, it is difficult to decide one ranking function corresponding to a single query category to rank documents for the query. However, our proposed approach does not rely on explicit query categorization at query time, and the learned ranking function can be used on a broad spectrum of queries.

For another reason, there exists a number of labeled doc-



(a) Navigational queries (b) Informational queries

Figure 7: Ranking performance (NDCG@K) of UQD-RankNet, SQD-RankNet, QC-RankNet and RankNet on Navigational/Informational queries of TREC2004

uments which are not critical for ranking in the training set, such as those should not be ranked in top positions. However, if this kind of documents are very difficult to rank, they may have much influence on the training process and attenuate the effect from important documents. By using query-dependent loss, we can filter out these hard-to-rank and unimportant documents from computing the risk of ranking so as to build a more effective ranking model from a healthier training process.

Furthermore, for query-dependent ranking function, it uses only a part of training dataset to learn the ranking model for each query category. It may cause the declining accuracy due to the lack of enough training examples. However, the approach of query-dependent loss can avoid the reduction in the number of training examples.

In addition, the shorter training and testing time is another advantage of query-dependent loss approach over query-dependent ranking approach. The training time of query-dependent ranking approach could be quite large, because many models need to be trained; while only one model is trained for query-dependent loss approach. And, at testing time, query-dependent ranking approach need to spend additional time in online searching the model which is best for each test query; while query-dependent loss approach use only one model. Therefore, it is more time-consuming to use query-dependent ranking approach than to use query-dependent loss approach.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed to incorporate query difference into constructing ranking models by introducing query-dependent loss functions. Based on a popular query taxonomy of Web search, we exploit the position-sensitive query-dependent loss function for ranking and apply it on two concrete ranking algorithms. Moreover, beyond the straightforward method learning ranking model with predefined query categorization, we have devised a new learning method which can conduct learning of the ranking model jointly with that of query categorization. Experimental results illustrate that the proposed query-dependent loss functions can significantly improves the accuracy of the learned ranking functions, and our new learning method achieve better performance than the straightforward method.

In the future, we plan to explore other potential approaches to incorporate query difference into learning to rank, especially we plan to investigate how to build query-dependent ranking models to serve the individual query. Furthermore, we will try to explore more meaningful query features and investigate their effects on both the accuracy of query classification and the performance of learned ranking functions.

7. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
[2] E. Baum and F. Wilczek. Supervised learning of probability distributions by neural networks. In *Proc. of NIPS*, 1987.
[3] S. Beitzel, E. Jensen, A. Chowdhury, and O. Frieder. Varying approaches to topical web query classification. In *Proc. of SIGIR*, 2007.
[4] A. Broder. A taxonomy of web search. *SIGIR Forum*, 2002.
[5] 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.
[6] 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.
[7] R. Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, 1997.
[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] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of ir techniques. In *ACM Transactions on Information Retrieval*, 2002.
[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of KDD*, 2002.
[12] I. Kang and G. Kim. Query type classification for web document retrieval. In *Proc. of SIGIR*, 2003.
[13] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proc. of SIGIR*, 2001.
[14] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Proc. of WWW*, 2005.
[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] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, 1998.
[18] S. Robertson. Overview of the okapi projects. In *Journal of Documentation*, 1998.
[19] D. Rose and D. Levinson. Understanding user goals in web search. In *Proc. of WWW*, 2004.
[20] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. In *Journal of ACM*, 1968.
[21] D. Shen, J. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proc. of SIGIR*, 2006.
[22] E. Voorhees and D. Harman. Trec: Experiment and evaluation in information retrieval. In *MIT Press*, 2005.
[23] F. Xia, T. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank - theory and algorithm. In *Proc. of ICML*, 2008.
[24] H. Zha, Z. Zheng, H. Fu, and G. Sun. Incorporating query difference for learning retrieval functions in information retrieval. In *Proc. CIKM*, 2006.
[25] C. Zhai, W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proc. of SIGIR*, 2003.
[26] 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.
[27] 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.