Learning the Gain Values and Discount Factors of Discounted Cumulative Gains

KE ZHOU, HONGYUAN ZHA, YI CHANG and GUI-RONG XUE

Abstract—Evaluation metric is an essential and integral part of a ranking system. In the past several evaluation metrics have been proposed in information retrieval and Web search, among them Discounted Cumulative Gain (DCG) has emerged as one that is widely adopted for evaluating the performance of ranking functions used in Web search. However, the two sets of parameters, the gain values and discount factors, used in DCG are usually determined in a rather ad-hoc way, and their impacts have not been carefully analyzed. In this paper we first show that DCG is generally not coherent, i.e., comparing the performance of ranking functions using DCG very much depends on the particular gain values and discount factors used. We then propose a novel methodology that can learn the gain values and discount factors from user preferences over rankings, modeled as a special case of learning linear utility functions. We also discuss how to extend our methods to handle tied preference pairs and how to explore active learning to reduce preference labeling. Numerical simulations illustrate the effectiveness of our proposed methods. Moreover, experiments are also conducted over a side-by-side comparison data set from a commercial search engine to validate the proposed methods on real-world data.

Index Terms—Discounted cumulative gains, evaluation metric, utility function, user preference, machine learning

1 Introduction

Evaluation metric is an essential component of any information retrieval system, because it directly influences the comparison of competing retrieval methods and ranking functions. Moreover, since evaluation metrics define what a desirable ranking should be, they have also been employed as the objective functions for directly optimizing ranking functions in the context of learning to rank [2]–[5].

To set the stage for our discussions, let us imagine a somewhat idealized use scenario of evaluation metrics: A Web search engine company has developed a new ranking function and would like to compare the performance of the new ranking function with the existing one used in production in order to make sure it indeed improves relevance of search results. To this end, a set of queries is sampled and the search results produced by both the new and old ranking functions are collected. Human editors are asked to judge the degree of relevance for each query-document pairs for the set of sampled queries. Finally, the evaluation metrics are computed based on the relevance judgments and the quality of the ranking functions is assessed using the evaluation metrics, e.g., if the new ranking function obtain higher scores according to the evaluation metrics than the old one, it can be the candidate pushed for release for production. This process clearly shows that evaluation metrics play a central role since they directly impact the evaluation results and thus the development and production decisions in Web search.

Several evaluation metrics are widely used in information retrieval and Web search including precision and recall, mean average precision (MAP) and discounted cumulative gains (DCG) [6]. Among those metrics, DCG has become quite popular for comparing the performance of ranking functions especially for Web search [7], [8]. In this paper, we focus on the critical issue of determining the gain and discount factors used in the definition of DCG. To this end, we first describe several technical notions related to DCG to facilitate our further discussions. We are interested in ranking N documents $\mathcal{X} = \{x_1, \ldots, x_N\}$. We assume that we have a finite ordinal label (grade) set $\mathcal{L} = \{\ell_1, \ldots, \ell_L\}$ with ℓ_i preferred over $\ell_{i+1}, i = 1, \ldots, L-1$.[1] In Web search, for example, we may have

$$\mathcal{L} = \{\text{Perfect}, \text{Excellent}, \text{Good}, \text{Fair}, \text{Bad}\},$$

i.e., $L = 5$. A ranking of $\mathcal{X}$ is a permutation

$$\pi = (\pi(1), \ldots, \pi(N)),$$

of $(1, \ldots, N)$, i.e., the rank of $x_{\pi(i)}$ under the ranking π is i. Furthermore, suppose that the label for the document x_i is labeled by the l_i-th label, where $l_i \in \{1, \ldots, L\}$. Then, $l_{\pi(i)}$ represents the label for the document at rank i under the permutation π.

Each ordinal label is associated with a *gain value* $g_i \equiv g(\ell_i)$, and $g_i, i = 1, \ldots, L$, constitute the set of gain values associated with $\mathcal{L}$. For example, the gain values for perfect, good and bad documents can be set to 10, 3 and 0, respectively. Intuitively,

1. This means that a document with label ℓ_i is considered more relevant than one with label ℓ_{i+1} with respect to the query in question.

This paper is an extended version of the workshop paper [1]. Here we provide detailed derivations of the theory as well as the proposed algorithm. We also consider the more practical case where the pair of rankings contain overlapping but possibly non-identical documents. In addition, we develop algorithms leveraging the idea of active learning for selecting ranking pairs for comparison. Last but not least, we evaluate the proposed method in a new data set from a commercial search engine, the results of which suggest the effectiveness of the method in real-world applications.

- *Ke Zhou and Hongyuan Zha are with the College of Computing, Georgia Institute of Technology, Atlanta, E-mail: {kzhou@, zha@cc.}gatech.edu*
- *Yi Chang is with Yahoo! Labs, Sunnyvale, CA, USA; email:yichang@yahoo-inc.com*
- *Gui-Rong Xue is with Alibaba Cloud Computing, Hangzhou, China*

Digital Object Indentifier 10.1109/TKDE.2012.252

the users gain more from a perfect document than from a bad document, and the gain values quantify the differences. In [7], the DCG for a ranking π with the associated labels, evaluated for its top K documents, is defined as

$$DCG_{g,K}(\pi) = \sum_{i=1}^{K} c_i g_{l_{\pi(i)}}, \quad K = 1, \ldots, N,$$

where $g_{l_{\pi(i)}}$ indicates the gain value for the rank i document $x_{\pi(i)}$; The set of parameters $c_1 > c_2 > \cdots > c_K > 0$ are the so-called *discount factors*. This reflects the fact that users emphasize more the documents on the top of the rankings.

Two important properties of DCG contribute to its popularity: 1) it can deal with multi-grade relevance judgments; and 2) it can also explicitly incorporate the position information of the documents in the result sets through the use of discount factors. Specifically, DCG as shown above is parametrized by two sets of parameters, the *gain values and discount factors*. Despite its widespread use in Web search and other applications, the choice of the two sets of parameters in DCG is mostly rather ad-hoc, and several different sets of values have been employed in various systems and evaluation experiments published in the literature — the parameters $g_i = 2^i - 1$ and $c_k = \frac{1}{\log(k+1)}$, for example, are quite common for Web search evaluations. Furthermore, it is unclear if those selected parameters are in anyway consistent with the general users' preference of rankings. This is rather an unsatisfactory situation considering the popularity and importance of DCG. In this paper, we try to fill the gap by addressing the following two important issues regarding the utilization of DCG as an evaluation metric:

1) Does the parameter set matter? I.e., do different parameter sets give rise to different preference over the rankings of the search results?
2) If the answer to the above question is *yes*, is there a principled way to select the set of the parameters?

It turns out that the answer to the first question is *yes* if there are more than two levels of relevance grade used in the evaluation.[2] This is discussed in [9] through experimental studies, but we will provide a more detailed analytic analysis. Specifically, this result implies that a new ranking function can appear to be better than an existing one when measured using one set of DCG parameters while possibly worse with another. It follows that in real applications, certain choices of the parameters may result in selecting the wrong ranking function for online service and thus may reduce the user experience in search engines. Moreover, as we will discuss later, different parameters may lead to very different evaluation results on the rankings that are close to each other. Thus, it is extremely important to use the right DCG parameters when validating small incremental improvements of ranking functions. Considering the fact that ranking functions are usually developed in a incremental and iterative way in commercial search engines, detecting small improvements of ranking functions can be quite a practical and vital task since the difference between the new and old ranking functions can be very small in this situation.

Given that the selection of DCG parameter has a great impact on the evaluation of search ranking functions, it is imperative that we select the parameters of DCG very carefully when evaluating information retrieval systems, since development and production decisions can very much depend on them. It is then quite natural to ask the second question: is there a principled way to select the DCG parameters? To address this question, we propose to *learn* the DCG parameters in a data-driven way rather than relying on ad-hoc manual selections. One advantage of this data-driven approach is that the parameters are directly determined by and will generally be consistent with the preferences of users, and it also allows for adaption to specific user sub-populations and system usage scenarios.

We will show that the DCG parameters can be viewed as the weights of a linear utility function and those associated weights can be estimated provided paired preferences over different rankings are available — Learning the weights can be then cast as an optimization problem that can be solved using convex quadratic programming very much like what is done in support vector machines for classification.

Experimentally, we carried out several numerical simulations that illustrate the feasibility and effectiveness of the proposed methodology. Additionally, a set of evaluation experiments are also conducted over a data set from a commercial search engine. The results of the experimental studies show that the proposed learning framework can estimate the parameters of DCG vectors accurately utilizing paired preferences over rankings. Moreover, the experiments also show that the proposed learning framework can significantly improve the accuracy of the comparisons when the quality of rankings are quite close to each other. To improve the effectiveness of the learning methods, we also address the problem of how to handle tied preference pairs and how to select ranking pairs for comparison through active learning.

The rest of the paper is organized as follows: In Section 2, we briefly discuss some related studies. Then, we show that DCG is not coherent in a very precise sense in Section 3. In Section 4, we present the proposed learning framework for estimating the parameters of DCG. The experiments over simulation data sets and a data set from a commercial search engine is presented in Section 5.4 where the issue of active learning is also addressed. We conclude our paper and discuss several future directions in Section 6.

2 Related Work

DCG as an evaluation metric was proposed in [7]. The results in [10] compare the sensitivity and reliability of several cumulative gain based metrics including DCG. In [11], several evaluation metrics are compared through their confidences in evaluations. All these studies focus on comparing the properties of different evaluation metrics. Despite the popularity of the cumulative gain based metrics, little research has been focused on analyzing the coherence of those metrics except the study of [9] which shows that different gain values of DCG can

2. This is generally the case for Web search where multiple grades are used to indicate the degree of relevance of documents with respect to a query.

give different judgements on the relevance of rankings. After the publication of [1], the problem of selecting gain values and discount factors is discussed in [12], but the focus of their work is to analyze the efficiency and stability introduced by sampling queries. A few recent work have studied whether evaluation metrics are consistent with the user preferences [13], [14]. Specifically, [14] suggests that the nDCG is the most effective evaluation metric among several other metrics. However, it is also shown in [14] that all evaluation metrics disagree with user preference to some extents, which implies that it is still important and useful to investigate methods that improve the evaluation metric in order to better capture user preferences. Recent development of evaluation metrics aims to capture more sophisticated requirement of users [15], [16]. For example, the work of [15] considers the diversity and novelty of the metric in addition to relevance. Similar to the case of DCG, the results of these evaluation metrics also largely rely on some parameters, which are usually set in a rather ad-hoc way. The learning framework proposed in this paper can be naturally applied to determine these parameters as well through learning different parameters for different types of queries or user intentions.

In parallel to the development of evaluation metrics, the learning to rank methodology has also attracted a lot of research interests in recent years. Several methods have been developed to learn the ranking functions through directly optimizing some performance metrics (or their approximate versions) such as MAP and DCG [2]–[4]. Our work can be considered as a natural component of the learning to rank framework in the following sense: 1) Learning to rank focuses on constructing a good ranking function with respect to some given performance metrics, while the goal of this paper is to analyze the coherence of DCG and propose a learning method to determine the parameters of DCG which will result in a more accurate metric for optimization for learning to rank. 2) Our proposed DCG parameter learning framework can be viewed as ranking different rank lists according to user preferences, and as such it is also an interesting application of the general learning to rank framework.

As will be seen in Section 4, DCG can be viewed as a linear utility function. Therefore, the problem of learning in DCG is closely related to the problem of learning utility functions which has been studied under the name of conjoint analysis in the marketing science community [17], [18]. The goal of conjoint analysis is to model the users' preferences over products and infers the features that satisfy the demands of users [19], [20].

3 Incoherency of DCG

We now focus on analyzing the properties of the gain values. Assume there are two rankers A and B using DCG with gain vectors g^A and g^B, respectively. We want to investigate how coherent A and B are in evaluating different rankings.

It can be observed from the definition that DCG takes both the relevance and position of the results into account. Specifically, the relevance of a result is captured by its gain value and the position is captured by the corresponding discount factor c_i. The gain vector $g = [g_1, \ldots, g_L]$ is said to be *compatible* if $g_1 > g_2 > \cdots > g_L$, reflecting the preferences over the given set of labels. If two gain vectors g^A and g^B are both compatible, then we say they are compatible with each other. In this case, there is a transformation ϕ such that $\phi(g_i^A) = g_i^B, i = 1, \ldots, L$, and the transformation ϕ is *order preserving*, i.e., $\phi(g_i) > \phi(g_j)$, if $g_i > g_j$.

3.1 Coherency of DCG with Binary Labels

In this section, we show that compatibility of the gain vectors implies coherency for a specific class of DCG metrics. Specifically, we show that DCG metrics with compatible gain vectors agree with each other when binary relevance labels are used, i.e., $L = 2$.

First we show that if A and B are compatible, then A and B agree on the set of rankings that are optimal, i.e., the set of rankings having the highest DCG. We first state the following well-known result.

Proposition 1: Let $a_1 \geq \cdots \geq a_N$ and $b_1 \geq \cdots \geq b_N$. Then

$$\sum_{i=1}^{N} a_i b_i = \max_{\pi} \sum_{i=1}^{N} a_i b_{\pi(i)}.$$

It follows from the above result that any ranking π with $g_{l_{\pi(1)}} \geq g_{l_{\pi(2)}} \geq \cdots \geq g_{l_{\pi(K)}}$ achieves the highest $DCG_{g,K}$, as long as the gain vector g is compatible. Therefore, any two rankers A and B using compatible gain vectors g^A and g^B, will score the above rankings with higher DCG values than any other rankings.

What about those rankings with smaller DCGs, i.e., non-optimal DCG values? We say two compatible rankers A and B are *coherent*, if they score any two rankings coherently, i.e., for rankings π_1 and π_2,

$$DCG_{g^A,K}(\pi_1) \geq DCG_{g^A,K}(\pi_2)$$

if and only if

$$DCG_{g^B,K}(\pi_1) \geq DCG_{g^B,K}(\pi_2),$$

i.e., ranker A considers π_1 is better than π_2 if and only if ranker B considers π_1 is better than π_2. In order to address the problem whether different DCG parameter sets give rise to different preference over the rankings, it is natural to ask if compatibility implies coherency? I.e., if rankers A and B have the same *order* of preferences for the labels, do they also agree on the preference of any pair of non-optimal rankings? We have the following result.

Theorem 1: If $L = 2$, then compatibility implies coherency.

Proof: Fix $K > 1$, and let $c = \sum_{i=1}^{K} c_i$. When there are only two labels, let the corresponding gains be g_1, g_2. For a ranking π, define

$$c_1(\pi) = \sum_{l_{\pi(i)}=1} c_i, \quad c_2(\pi) = c - c_1(\pi).$$

Then $DCG_{g,K}(\pi) = c_1(\pi)g_1 + c_2(\pi)g_2$. For any two rankings π_1 and π_2,

$$DCG_{g^A,K}(\pi_1) \geq DCG_{g^A,K}(\pi_2)$$

implies that

$$c_1(\pi_1)g_1^A + c_2(\pi_1)g_2^A > c_1(\pi_2)g_1^A + c_2(\pi_2)g_2^A$$

which gives

$$(c_1(\pi_1) - c_1(\pi_2))(g_1^A - g_2^A) > 0.$$

Since A and B are compatible, the above implies that

$$(c_1(\pi_1) - c_1(\pi_2))(g_1^B - g_2^B) > 0.$$

Therefore $DCG_{g^B,K}(\pi_1) \geq DCG_{g^B,K}(\pi_2)$. The proof is completed by exchange A and B in the above arguments. □

3.2 Incoherency of DCG when $L > 2$

Not too surprisingly, compatibility does not imply coherency when $L > 2$, which is illustrated in the following example.

Example 1: Let $\mathcal{X} = \{x_1, x_2, x_3\}$, i.e., $N = 3$. We consider $DCG_{g,K}$ with $K = 2$. Assume the labels of x_1, x_2, x_3 are ℓ_2, ℓ_1, ℓ_3, and for ranker A, the corresponding gains are $2, 3, 1/2$. The optimal ranking is $(2,1,3)$. Consider the following two rankings, $\pi_1 = (1,3,2)$, and $\pi_2 = (3,2,1)$. None of them is optimal. Let the discount factors be

$$c_1 = 1 + \varepsilon, \quad c_2 = 1 - \varepsilon, \quad 1/4 < \varepsilon < 1.$$

It is easy to check that

$$DCG_{g^A,2}(\pi_1) = 2c_1 + (1/2)c_2$$
$$> (1/2)c_1 + 3c_2 = DCG_{g^A,2}(\pi_2).$$

Now let $g^B = \phi(g^A)$, where $\phi(t) = t^k$, and ϕ is certainly order preserving, i.e., A and B are compatible. However, it is easy to see that for k large enough, we have

$$2^k c_1 + (1/2)^k c_2 < (1/2)^k c_1 + 3^k c_2$$

which is the same as $DCG_{g^B,2}(\pi_1) < DCG_{g^B,2}(\pi_2)$. Therefore, A thinks π_1 is better than π_2 while B thinks π_2 is better than π_1 even though A and B are compatible. This implies A and B are *not* coherent.

REMARKS. When we have more than two labels, which is generally the case for Web search, using DCG_K with $K > 1$ to compare the DCGs of various ranking functions will very much depend on the gain vectors used. Different gain vectors can lead to completely different conclusions about the performance of the ranking functions. Similar conclusions can be drawn for the discount factors. Moreover, for variations for DCG such as NDCG, the similar analysis can be conducted to show that they are not coherent in general, which implies that the incoherency of DCG can be viewed as the core problem of these metrics.

We now further illustrate this result by investigating the possible DCG values generated with two different sets of parameters: We generate several rankings by randomly permuting the ranking list of ten documents with the following relevance labels

$$[5,5,4,4,3,3,2,2,1,1].$$

For each pair of the generated rankings, we compute the DCGs using two different sets of gain values that are widely used in the evaluation of search engines: $g_l = 2^l - 1$ and $g'_l = l$ while we use the same discount factors $c_i = 1/\log(i+1)$. Our goal is to investigate the fraction of pairs that are agreed by the two different DCG metrics. In Figure 1, we show the fraction of agreed pairs with respect to the DCG differences under gain values g_l. Specifically, we use g_l and c_i as DCG parameters to compute the DCG values for each rankings and measure the different between each pair of rankings. We can see in Figure 1 that different gain values do not agree with each other in general. We can see that the fraction of agreed pairs are very small if the DCG difference using gain values g is small. However, even when the DCG difference using gain values g is 1, there are still only about 80% pairs agreed by both metrics — This indicates that DCG with gain values g_l and g'_l contradicts with each other on a substantial fraction of pairs even when their DCG differences considered by the ranker using gain values g are quite large. This is to say that for a substantial fraction of ranking pairs, ranker with gain values g says one ranking is better than the other in the pair with high confidence, but the ranker with gain values g' says the opposite is true. Therefore, we conclude that different parameters of DCG can impact comparison of rankings to a great extent and thus should be selected in a more principled approach rather than heuristically.

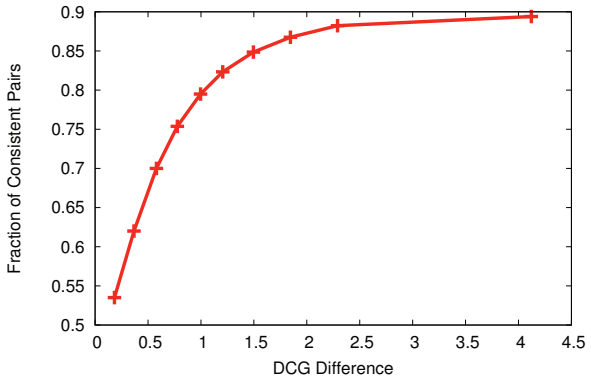


Fig. 1. The fraction of pairs that are consistent under two different set of DCG parameters. The x-axis shows the DCG differences under gain value $g_l = 2^l - 1$. The y-axis represents the fraction of ranking pairs judged consistently by two metrics.

4 Learning Gain Values and Discount Factors

The conclusion of the incoherency of DCG in Section 3 implies that the values of the parameters used in DCG can have a great impact on the evaluation of information retrieval systems. This is also supported by the experimental studies in [9]. Thus, these parameters should be determined in a principled and systematic approach that could capture the users' preferences over search results. In this section, we propose a learning framework to determine the parameters of DCG based on paired preferences over rankings. The main idea of our approach is to consider DCG as a simple form of a linear *utility function*. In this section, we make this statement

more precise and discuss a method to learn the gain values and discount factors of DCG that constitute this utility function.

4.1 Utility Functions

From the perspective of economics, search engines satisfies the information need of users described by queries through providing ranked search results to them. Therefore, the quality of a ranking list should be measured by the degree of satisfaction of users. Here, we employ the concept of *utility functions* to quantify the satisfaction of users.

Specifically, every evaluation metric can be viewed as a utility function $u : \Pi \to \mathbb{R}$ that assigns a real number $u(\pi)$ for each ranking list π of a given query. Larger values of $u(\pi)$ implies that higher degree of relevance, i.e., $u(\pi_1) > u(\pi_2)$ if the ranking list π_1 is preferred to π_2 for a given query.

4.2 A Binary Representation

Before describing our learning framework, we first introduce a binary representation format for a given ranking list, which leads to more convenient and simple formulation of the utility function. We consider a fixed K, and we use a binary vector $s(\pi)$ of dimension KL, i.e., the product of K and L, to represent a ranking π considered for $DCG_{g,K}$. Here L is the number of levels of the labels. Particularly, the first L components of s correspond to the first position of the K-position ranking in question, and the second L components the second position, and so on. Within each L components, the i-th component is 1 if and only if the document in position one has label $\ell_i, i = 1, \ldots, L$.

Example 2: In the Web search case with $L = 5$ labels, suppose we consider $DCG_{g,3}$, and for a particular ranking π the labels of the first three documents are

`Perfect, Bad, Good`.

Then the corresponding 15-dimensional binary vector

$$s(\pi) = [1,0,0,0,0,\ 0,0,0,0,1,\ 0,0,1,0,0].$$

With the above binary representation of rankings, we postulate a utility function $u(\pi) = w^T s(\pi)$ which is a linear function of $s(\pi)$, and w is the weight vector, to measure the quality of $s(\pi)$. We write

$$w = [w_{1,1}, \ldots, w_{1,L},\ w_{2,1}, \ldots, w_{2,L},\ \ldots, w_{k,1}, \ldots, w_{K,L}].$$

We often use the simplified notation $u(s) = w^T s$ where the binary vector s uniquely determines the ranking for the input of the utility function. We now distinguish two cases.

CASE 1. The gain values are position *independent*. This corresponds to the case

$$w_{i,j} = c_i g_j, \quad i = 1, \ldots, K, j = 1, \ldots, L.$$

This is to say that $c_i, i = 1, \ldots, K$ are the discount factors, and $g_j, j = 1, \ldots, L$ are the gain values. It is easy to see that

$$w^T s(\pi) = DCG_{g,K}(\pi).$$

Therefore, the widely used DCG can be viewed as a special case of the linear utility function.

CASE 2. In this framework, we can consider the more general case that the gain values are *position dependent*. Then $w_{1,1}, \ldots, w_{1,L}$ are just the products of the discount factor c_1 and the position dependent gain values for position one, two, and so on. In this case, there is no need to separate the gain values and the discount factors. The weights in the weight vector w are what we need.

4.3 Learning the Weight Vector w

Assume that we have available a partial set of preferences over the set of all rankings. For example, we can present a pair of rankings π_1 and π_2 to a user, and the user prefers π_1 over π_2, denoted by $\pi_1 \succ \pi_2$, which translates into $w^T s(\pi_1) \geq w^T s(\pi_2)$ in terms of the utility function $u(\pi)$. Let the set of available preferences be

$$\pi_i \succ \pi_j, \quad (i, j) \in S.$$

Our goal is to construct a utility function $u(\pi)$ that is as consistent as possible with the given set of preferences.

Let us first discuss the second case described above, we can formulate the learning problem as estimating the weight vector w subject to a set of preference constraints. The formulation can be considered as a variation of RankSVM [21] with additional constraints that are specific for the utility function learning problem.

$$\min_{w,\, \xi_{ij}} w^T w + C \sum_{(i,j) \in S} \xi_{ij}^2 \tag{1}$$

subject to

$$w^T (s(\pi_i) - s(\pi_j)) \geq 1 - \xi_{ij},\ \xi_{ij} \geq 0,\ (i, j) \in S. \tag{2}$$

$$w_{kl} \geq w_{k,l+1},\ k = 1, \ldots, K,\ l = 1, \ldots, L-1. \tag{3}$$

Here ξ_{ij} are the so-called slack variables to account for possible errors in preference data. The coefficient C of the second term is a regularization parameter that controls the trade off between fitting the preference data and model complexity. Usually, the value of C can be determined by *cross validation*. As an alternative, several algorithms are proposed for calculating the regularization path for SVM models which can also be used to determine C [22].

The constrains in Equation (2) are used to enforce the requirement that the weight vector w is consistent with the known preference pairs, which is similar to the set of constraints used in RankSVM. Additionally, the constrains in Equation 3 encode the fact that replace a document with a more relevant one will increase the value of utility function.

The above optimization problem is convex and thus can be solved by a lot of solvers. In our implementation, we make use of the CVX[3] which is an off-the-shelf convex programming solver. The time complexity of the solving the problem is polynomial and depends on the specific optimization tools used by CVX. In general, the solver runs very fast in all our data sets (less than 5 minutes for the largest data we consider).

For the first case discussed above, we can compute w as in Case 2, and then find c_i and g_j to fit w.[4] Particularly, we

3. http://cvxr.com/

4. It is also possible to carry out hypothesis testing to see if the gain values are position dependent or not.

can reconstruct the gain vector and the discount factors from the estimated $\hat{w}$. To this end, we rewrite $\hat{w}$ as a matrix W of size $L \times K$. Assume that singular value decomposition of the matrix W can be expressed as $W = U\text{diag}(\sigma_1, \ldots, \sigma_n)V^T$ where $\sigma_1 \geq \cdots \geq \sigma_n$. Then, the rank-1 approximation of W is $\hat{W} = \sigma_1 u_1 v_1^T$. In this case, the first left singular vector u_1 is the estimation of the gain vector and the first right singular vector v_1 is the estimation of the discount factors.

5 Experiments

In this section, we present several experiments to evaluate the proposed learning framework for DCG parameters. The experimental studies are conducted over both simulation data sets and a real-world data set from a commercial search engine.

5.1 Evaluation Measures

Given the estimated $\hat{w}$ and the ground-truth w, we apply two measures to evaluate the quality of the estimated $\hat{w}$. 1) The first measure is the precision on a test set. A number of pairs of rankings are sampled from the set of all rankings as the test set. We apply $\hat{w}$ to predict the preference over the test set. Then the precision of $\hat{w}$ is calculated as the proportion of correctly predicted preference (based on the true w) in the test set. 2) The second measure is the similarity of w and $\hat{w}$. Given the true value of w and the estimation $\hat{w}$ defined by the above optimization problem, a straightforward similarity measure is the cosine similarity between them. However, cosine similarity can be influenced by adding constant offsets to the weight vectors. Specifically, the cosine similarity between $w_1 + a$ and $w_2 + a$ can be very high even if w_1 and w_2 are quite different when the constant vector a is quite large. In order to get around this, we define the transformation $T(w)$ as follows:

$$\begin{aligned} T(w) =& (w_{11} - w_{1L}, , \ldots, w_{1L} - w_{1L}, \\ & \ldots, w_{K1} - w_{KL}, \ldots, w_{KL} - w_{KL}) \end{aligned} \quad (4)$$

We can observe that the transformation T rules out the effect of the constant offset of w which does not impact the comparison two rankings and thus it preserves the orders between rankings, i.e., $T(w)^T s(\pi_1) > T(w)^T s(\pi_2)$ iff $w^T s(\pi_1) > w^T s(\pi_2)$. Then, we measure the similarity between w and $\hat{w}$ by $\text{sim}(w, \hat{w}) = \frac{T(w)^T T(\hat{w})}{\|T(w)\| \|T(\hat{w})\|}$, which represents the cosine similarity between the two transformed weight vectors.

5.2 Simulation Studies

In this section, we report the results of numerical simulations to show the feasibility and effectiveness of the method proposed in Equation (1). We call this method DCGLearn for the sake of convenience.

5.2.1 Experimental Settings

We use a ground-truth w to obtain a set of paired preferences of rankings. Our goal is to investigate whether we can reconstruct the unknown w via learning from the given set of paired preferences. The ground-truth w is generated as

$$w_{kl} = g_l / \log(k+1) \quad (5)$$

For a comprehensive comparison, we distinguish two settings of $g_l, l = 1, \ldots, L$ where $L = 5$. Specifically, we set $g_l = l$ in the first setting (Data 1), and $g_l = 2^l - 1$ in the second setting (Data 2). These two set of parameters are widely used for evaluating information retrieval systems.

The rankings are obtained by randomly permuting a ground-truth ranking list. For example, the rankings can be generated by randomly permuting the list of length 10 (K=10)

$$[5, 5, 4, 4, 3, 3, 2, 2, 1, 1].$$

We generate a collection of pairs of rankings and use the ground-truth w to judge which ranking is preferred. Specifically, if $w^T s(\pi_1) > w^T s(\pi_2)$, we have a preference pair $\pi_1 \succ \pi_2$. Otherwise we have a preference pair $\pi_2 \succ \pi_1$.

5.2.2 General Performance

We randomly sample a number of rankings and generate preference pairs according to a ground-truth w. The number of preference pairs in training set varies from 20 to 800. We plot the prediction accuracy on test set of the estimated $\hat{w}$ with respect to the number of training pairs in Figure 2 and Figure 3, respectively.

It can be observed from Figure 2 and 3 that the performance of DCGLearn generally improves with the increasing of training pairs, indicating that the preferences over rankings can be utilized to enhance the estimation of the unity function w. Another observation is when about 800 preference pairs are included in training set, the precisions in test sets become close to 98% under both settings, which is quite close to the ground-truth utility function. This observation suggests that we can estimate w very accurately from the preferences of rankings.

We also show a baseline Other that use the ground-truth utility function from the wrong data set, i.e., we use the ground-truth measure of Data 2 to predict the preference pairs of Data 1 and vice visa. The prediction accuracy of this baseline is less than 90%, which indicates that the two different ground-truth utility function disagrees with each other on more than 10% of the pairs. This can be a quite large fraction of pairs in the evaluation of information retrieval systems, which may lead to completely different conclusions on performance comparisons of systems. In order to investigate this point more closely, we show the prediction accuracy with respect to the DCG difference in Figure 4 and Figure 5. From those figures, we can observe that DCGLearn outperforms the baseline with a large margin when DCG differences of pairs are relatively small. This is because the pairs of rankings that are relatively similar to each other are more difficult to tell apart, and thus are more sensitive to parameters in the DCG metrics used. This has important implications in practice, because the quality of ranking functions used in real world information retrieval systems are usually quite close to each other with overall average DCGs differ by a few percentage points. Consequently, it is important to use the right parameters for DCG when comparing their performance.

We also measure the similarity between the learned weight vector and the ground-truth as defined in Section 5.1. The similarity with respect to the number of training pairs is shown

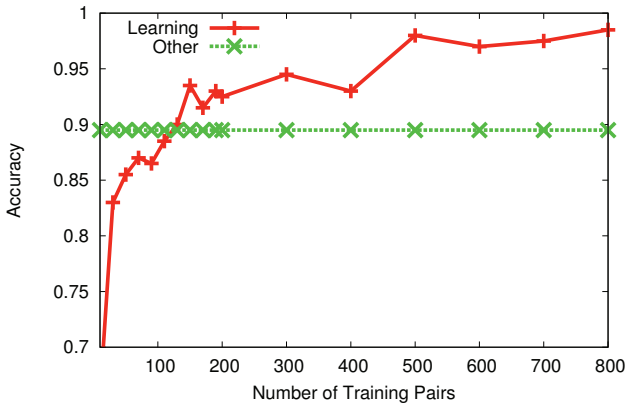


Fig. 2. Prediction accuracy over the test set with respect to the number of training pairs on Data 1

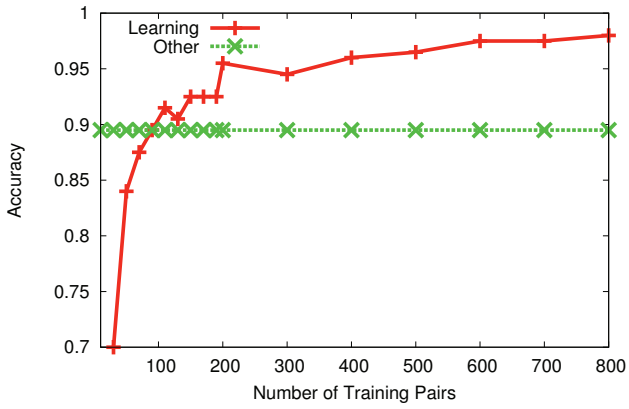


Fig. 3. Prediction accuracy over the test set with respect to the number of training pairs on Data 2

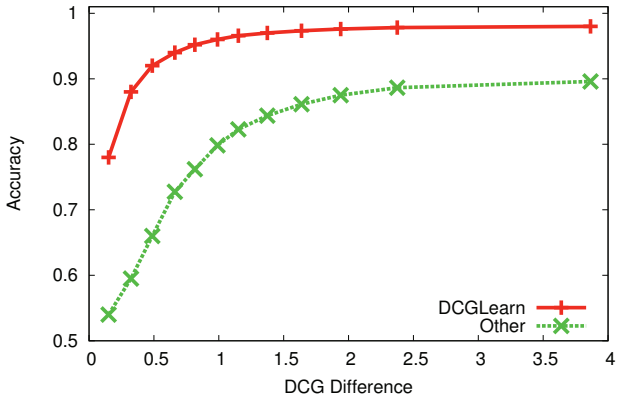


Fig. 4. Prediction accuracy over different subsets of the test set of Data 1

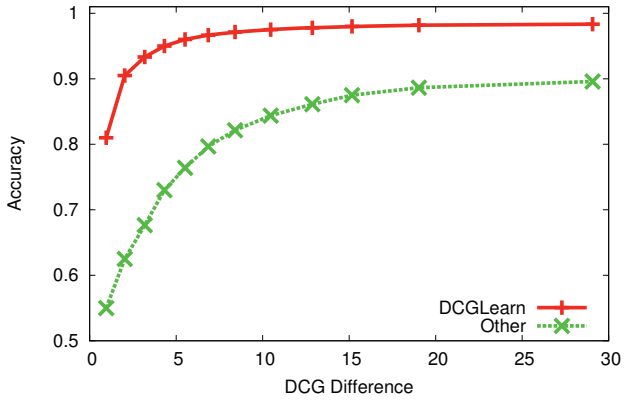


Fig. 5. Prediction accuracy over different subsets of the test set of Data 2

in Figure 6. We notice that the similarity and precision sometimes give different conclusions of the relative performance over Data 1 and Data 2. We think it is because the similarity measure is sensitive to the choice of the offset constant. For example, large offset constants will give similarity very close to 1. Currently, we use $w_{1L}, \ldots, w_{KL}$ of the offset constant as in Equation (4). Generally, we prefer the prediction accuracy as a more meaningful evaluation measure and report similarity as a complementary measure.

5.2.3 Noisy Settings

In real-world scenarios, the preference pairs of rankings can be noisy. Therefore, it is interesting to investigate the effect of the errors in the preference pairs to the performance of the learned DCG metrics. To this end, we fix the number of training pairs to be 800 and create the noisy pairs by randomly reverse the preference of several pairs in the training set. In our experiments, the number of noisy pairs ranges from 5 to 200. Since the trade off value C is important to the performance in the noisy setting, we select the value of C that shows the best performance on an independent validation set. We report the prediction accuracy with respect to the number of noisy pairs in Figure 7. We can observe that the performance decreases *gracefully* when the number of noisy pairs grows. However,

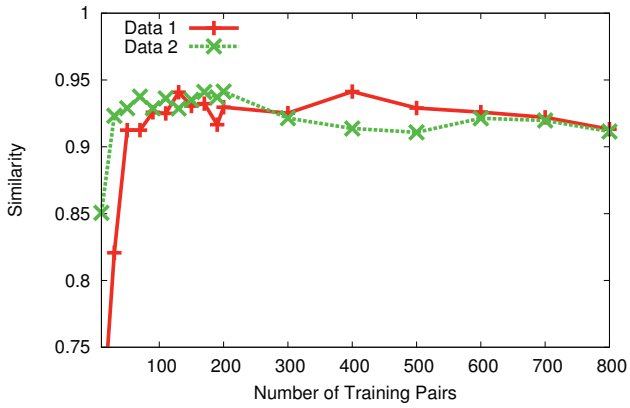


Fig. 6. Similarity between w and $\hat{w}$ with respect to the number of training pairs on Data 1 and Data 2

the degradation in performance is graceful, and with about 25% noisy pairs, the prediction accuracy is still about 85%.

In addition to the experiments using noisy preference pairs, we also consider errors in the relevance grades of documents. In this case, we modify the grades of a number of documents to be a random grade between 1 and 5 and thus introduce

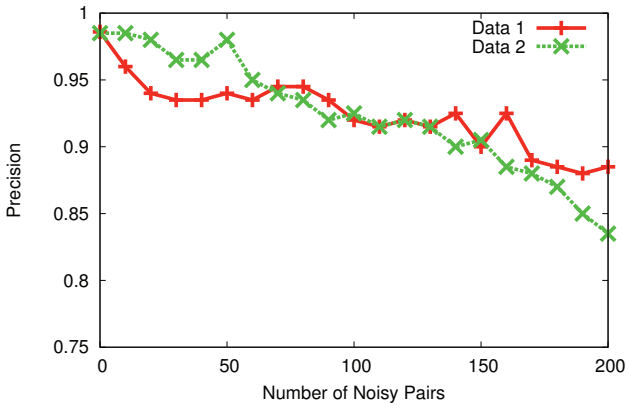


Fig. 7. Precision over the test set with respect to the number of noisy pairs in noisy settings

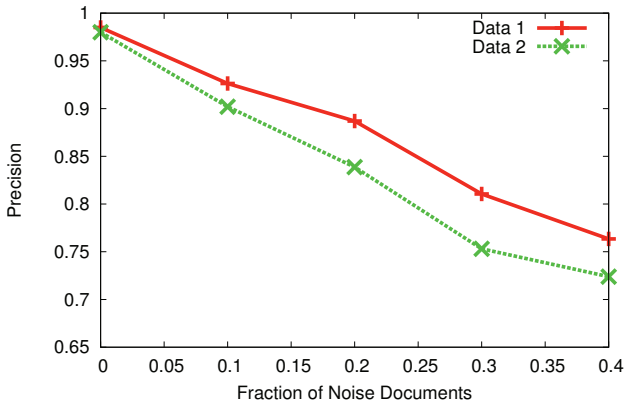


Fig. 8. Precision over the test set with respect to the number of noisy grades in noisy settings

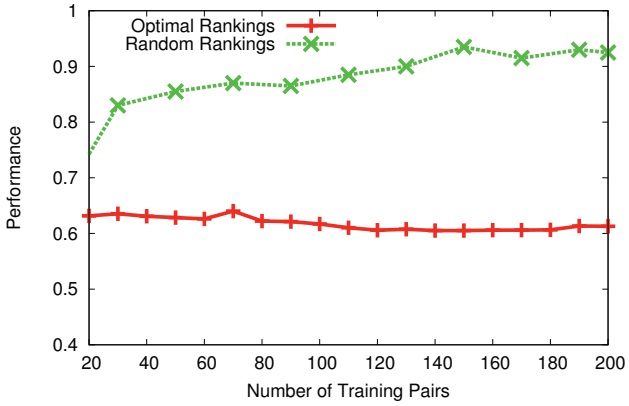


Fig. 9. Performance when training pairs are generated by permuting the same optimal rankings

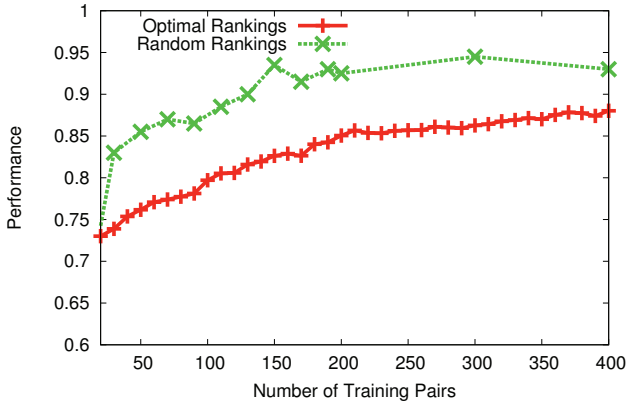


Fig. 10. Performance when training pairs are generated from different lists

noise in the training set. The estimated $\hat{w}$ is used to predict the preferences on a test set. The precision with respect to the number of noisy documents is shown in Figure 8. It can be observed that the performance decreases when the number of noisy documents increases, but again the degradation in performance is graceful, similar to the case of errors in preferences.

5.2.4 An Experiment on Using Optimal Rankings

In Proposition 1 of Section 3, we prove that all the DCG parameters agrees on the optimal rankings. We further investigate the impact on optimal rankings empirically.

First, in order to verify Proposition 1 empirically, we restrict the preference pairs by involving the optimal ranking in each pair of training data. For example, we set one ranking of the preference pair to be the optimal ranking $[5,5,4,4,3,3,2,2,1,1]$ and the other ranking is obtained by permuting the optimal ranking randomly. In this case, if the other ranking is generated by permuting the same list, it is implied by Proposition 1 that any compatible gain vectors will agree on the fact that the optimal ranking is preferred to other rankings. The preference pairs do not carry any constraints to the utility function w. Therefore, the constraints corresponding to these preference pairs are not effective in determining the utility function w. This conclusion is verified empirically as shown in Figure 9 that the performance does not increase when the number of training pair grows.

Now we can consider another setting: As in the first setting, we set one ranking of the preference pair to be the optimal ranking $[5,5,4,4,3,3,2,2,1,1]$. However, the other ranking is obtained by permuting a different set of grades, e.g., $[5,5,5,4,4,4,3,3,2,1]$. Thus, a fraction of the constraints induced from the pairs can be effective to determine the utility function. In Figure 10, we show the performance measured by accuracy on test set with respect to the number of training pairs. We also show the results for random rankings as we did in Section 5.2.2 (labeled by Random Rankings). We can observe that when the type of preferences is restricted, the learning algorithm requires more pairs to obtain a comparable performance. We conclude from this observation that some pairs are more effective than others to determine w. Thus, if we can design algorithms to select these more effective preference pairs, the number of pairs required for training can be greatly reduced. We will continue this line of ideas in Section 5.5.

5.3 Simulations Based on Microsoft Learning to Rank Data

In Section 5.2, we assume that the rankings are generated from the permutations of $[5,5,4,4,3,3,2,2,1,1]$. One drawback of this assumption is that it ignores the fact that the grade distributions for different search result sets can be quite different in practice. For example, for navigational queries, there is usually a single results with grade 5 and the grades for all the other results are much lower. While informational queries may have quite different grade distribution with several documents labeled as 4 or 5. In this section, we consider real-world search result sets by exploring the Microsoft Learning to Rank Data[5], which are generated from a commercial search engine. There are two data sets Web10K and Web30K with 10,000 and 30,000 queries, respectively. The five-level relevance judgments are provided to measure the degree of relevance for each query-document pair in the data sets.

As in Section 5.2, we use the following utility function as the ground-truth to generate the preferences between rankings,

$$w_{kl} = \frac{2^l - 1}{\log(k+1)}.$$

For each query, we generate random rankings by permuting all its corresponding documents and take the first ten documents in the resulting ranking list. For each pair of rankings corresponds to the same query, the ground-truth measure is used to determine which ranking list is preferred.

We perform five-fold *cross-validation* and report the average accuracy over the five folds. In Figure 11, we show the average accuracy with respect to the number of the training pairs. It can be observed that the performance increases when the number of training pairs grows. The overall performance is almost the same as the simulations we performed in the previous section, which indicates that the proposed framework works quite well when the search results come from real-world search engines. Similarly, we include the baseline Other as described in previous section, which uses a different set of parameters of DCG $w_{kl} = \frac{l}{\log(k+1)}$. We can see that this baseline achieves 90% accuracy, which is similar to the results for simulations in Section 5.2. Again, by splitting the test set according to the DCG difference of the ranking pairs, we can see that DCGLearn works much better than the baseline when the DCGs of the ranking pairs are close to each other; this again parallels our observations made in Figure 4 and Figure 5. Thus, we can conclude that DCGLearn can capture the subtle differences in the quality of rankings, which can be important when comparing ranking functions that perform close to each other.

We also perform experiments to evaluate the proposed framework when there are noises in the relevance judgements. In real-world scenarios, relevance judgements for query-document pairs are usually made by human editors. Their understandings of query intentions may be different with search engine users. In order to evaluate our method on the case of noisy relevance judgements, we sample a fraction of documents for each query and modify their relevance judgement by choosing a grade between 1 and 5 randomly. The performance with respect to the fraction of noisy judgements are presented in Figure 14 and 15 for Web10K and Web30K data set, respectively. It can be observed that the prediction accuracy decreases when the fraction of noisy judgement increases. The degradation in performance is again graceful. Also, We report the performance with 200, 600, 1000 training pairs for each dataset. We can see that more training pairs improves the performance in general as in the noiseless case.

5. http://research.microsoft.com/en-us/projects/mslr/

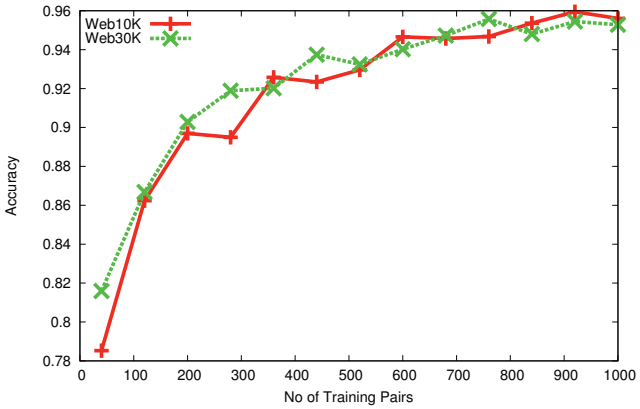


Fig. 11. Prediction accuracy with respect to the number of training pairs on Web10K and Web30K data set

TABLE 1
The description of side-by-side judgements from editors

Label	Description	Number in data set
1	Set one much better	13
2	Set one better	108
3	Both sets the same	473
4	Set two better	148
5	Set two much better	24

5.4 Experiments on News Search Data

In order to evaluate the proposed learning methodology using more realistic settings, we also conduct experiments on a preference data set from a commercial news search engine. We collect the top five results of two different rankers for 766 queries randomly sampled from the search logs. Human editors are asked to label the search results with four-level of grades according to their relevance to the queries. In particular, these labels are used to evaluate the relevance of ranking functions in the search engine, but with a set of parameters that are set heuristically. The preference data of rankings are labeled by human editors in a side-by-side setting. Specifically, we present the rankings produced by two rankers to the human editors with random order and ask them to choose grades from 1-5 which represents her degree of preference on the two rankings. The meaning of the grades is described in Table 1 in detail.

From Table 1, we observe that the results produced by the two rankings are of the same degree of relevance (labeled by 3) for a large fraction of queries. We remove these pairs from the data set and use all the remained pairs for evaluating

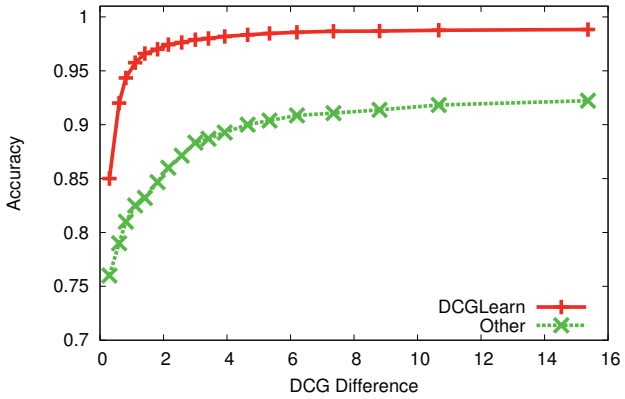


Fig. 12. Prediction accuracy over different subsets of the test set of Web10K

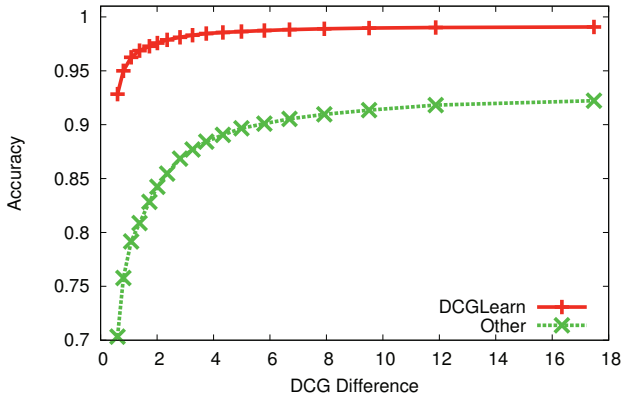


Fig. 13. Prediction accuracy over different subsets of the test set of Web30K

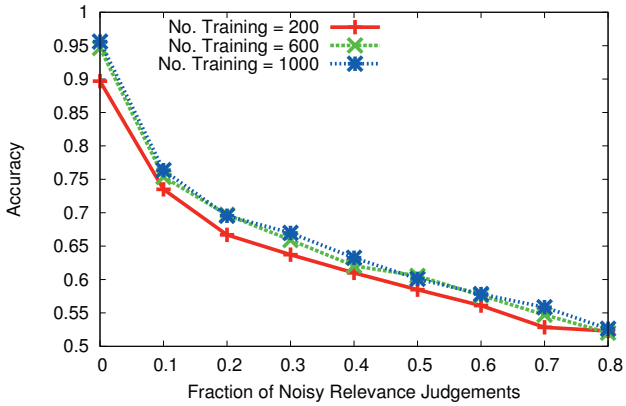


Fig. 14. Prediction accuracy with respect to the fraction of noisy pairs in training set on Web10K data set. The plot shows performance with 200, 600, 1000 training pairs.

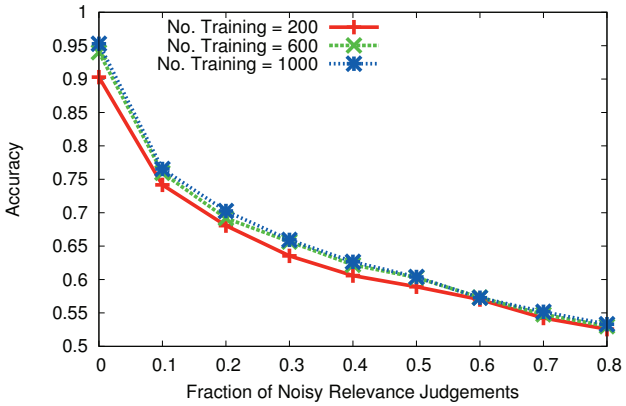


Fig. 15. Prediction accuracy with respect to the fraction of noisy pairs in training set on Web30K data set. The plot shows performance with 200, 600, 1000 training pairs.

our method. Ideally, a good DCG metric should be consistent with the side-by-side judgements of the rankings. Thus, we can use the prediction accuracy to measure the effectiveness the learned DCG parameters. In order to evaluate the proposed learning framework on the news data set. We perform 5-fold *cross validation* on the data set. For each fold, we use 80% data to learn the utility function with DCGLearn and report the prediction accuracy on the rest 20% data. Moreover, the differences between rankings labeled by 1 or 5 are more significant than those labeled by 2 or 4 as we can observed from Table 1. For the sake of convenience, we denote the set of pairs in test set labeled by 1 or 5 by Set1 and the set of pairs labeled by 2 or 4 by Set2. We report the prediction accuracy on the whole test set as well as on Set1 and Set2 on Figure 16. As baselines, we consider two sets of gain values and discount factors that are commonly used to compare the performance of ranking functions:

- Baseline 1.

$$g_l = 2^l - 1, \quad c_k = \frac{1}{\log(k+1)}$$

- Baseline 2.

$$g_l = l, \quad c_k = \frac{1}{\log(k+1)}$$

We also include a naive baseline Random that predicts the preference on the rankings randomly.

From Figure 16, all Baseline1, Baseline2 and DCGLearn substantially outperform Random. Thus, both the heuristically selected and learned DCG parameters can captures the preferences on rankings to some extent. More importantly, we observe that the parameters obtained by DCGLearn can predict the preferences more accurately than both Baseline1 and Baseline2 on both Set1 and Set2. Thus, the parameters obtained by DCGLearn is more effective than the two sets of heuristic DCG parameters. Moreover, the prediction accuracy for Set1 is higher than that of Set2. This is expected since Set1 contains the pairs labeled by 1 or 5 which indicates the one ranking is much better than the other. Thus, editors have clear preference on one ranking over the alternative in Set1. We note that DCGLearn can improve the performance on Set1, which indicates that: 1) There is relatively significant differences between the preferences of the human editors

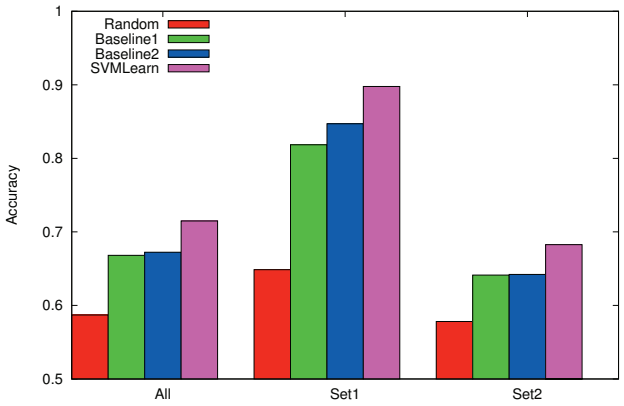


Fig. 16. Prediction accuracy of Random, Baseline1, Baseline2 and the parameters learnt from training data on the whole test set, Set1 and Set2

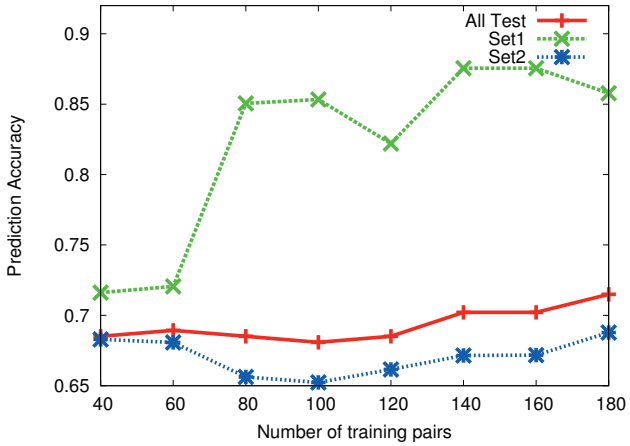


Fig. 17. Prediction accuracy with respect to the number of training pairs on the whole test set, Set1 and Set2

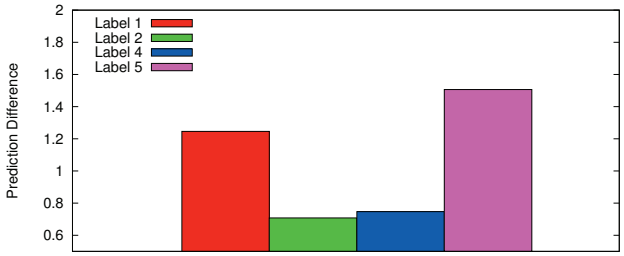


Fig. 18. Prediction differences for different labels

and those captured by the commonly used DCG parameters, implying that DCG with heuristic parameters is generally not an accurate measure to evaluate relevance of rankings. 2) DCGLearn can be used to reduce the gap between the preference of humans and the commonly used DCG parameters. Thus, DCGLearn can be used to find DCG parameters more consistent with user preferences. On the other hand, the prediction accuracy on Set2 is generally much lower since the difference in relevance for pairs in Set2 can be quite subtle even to human editors. The performance improvement on Set2 shows that DCGLearn can better capture these subtle differences of relevance for rankings. Moreover, we also plot the averaged absolute differences for pairs with different judgements in Figure 18. We can see that the prediction difference is significantly larger for pairs that are labeled by 1 or 5 than those are labeled by 2 or 4. This observation suggests that the learnt weight vector can not only predict the right order for pairs but also indicate the degree of their differences.

We plot the prediction accuracy with respect to the number of training pairs in Figure 17. As mentioned before, we plot both the performance on Set1, Set2 and all test set. We observe that the performance increases as the number of training pairs grows in general.

Our training set contains two types of pairs: 1) pairs labeled by 1 or 5, which indicate that the relevance difference are quite large and 2) pairs labeled by 2 or 4, representing relatively small difference in relevance. In order to investigate the potential different degree of impacts of those two types of ranking pairs, we perform experiments based on training data including one type of pairs only. The results are shown in Figure 19. As in previous experiments, we report the prediction accuracy on Set1, Set2 and all the test set.

We observe the prediction accuracy on all test pairs reduces when using only subset of the training data. This is because that the learnt model is not accurate when the training data is not sufficient. An interesting observation is that the parameters learned from ranking pairs labeled by 1 and 5 performs better on Set1 although its performance on the other two test set is not very good. This can be explained by the fact the ranking pairs labeled by 1 or 5 have a different distribution with those labeled by 2 or 4. On the other hand, we observe that making use of all training data archives the best performance in all cases, indicating that the amount of training data is important to learn better DCG parameters.

5.4.1 Modeling Tied Pairs

In Table 1, we can observe that there is substantial amount of tied ranking pairs in real-world preference data, i.e., pairs that are of the same degree of relevance judged by the human editors (labeled by 3). We note that those tied pairs also captures important information about the relevance of the ranking lists. In real application, we should not only identify ranking functions that improves the relevance of search results, but prevent modifications that do not improve the user satisfactory as well. Therefore, it is also desirable to take in account the tied pairs in the learning process of the DCG parameters.

To this end, assume that in addition to the preference over ranking list $\{\pi_i \succ \pi_j, (i,j) \in S\}$, we also have a set of tied pairs over ranking lists:

$$\{\pi_i = \pi_j, \quad (i,j) \in T\}.$$

In this case, we seek a weight vector that is consistent with both types of pairs. Thus, we have the following optimization

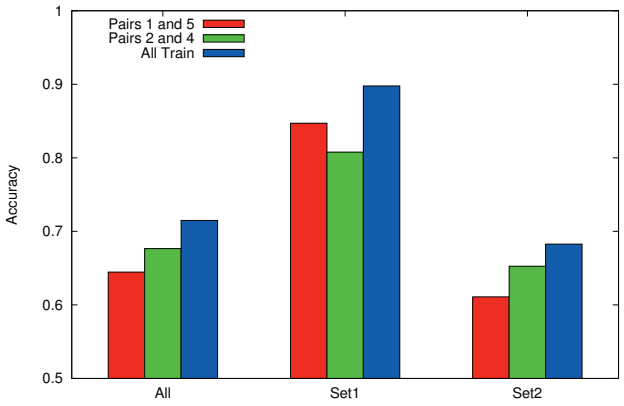


Fig. 19. The impact of different subsets of training pairs. We show the prediction accuracy on the whole test set, Set1 and Set2

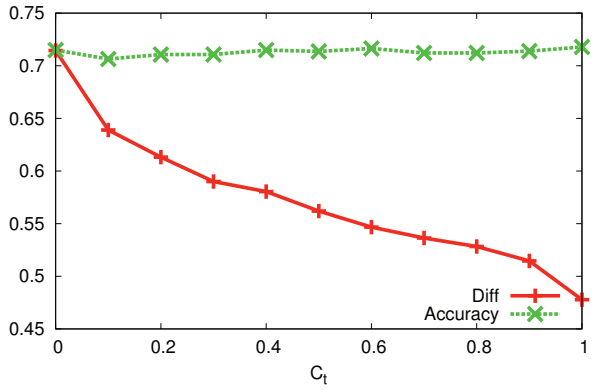


Fig. 20. Modeling tied pairs: The performance measured by accuracy on the test set and the averaged score differences

problem:

$$\min_{w,\,\xi_{ij}} w^T w + C_s \sum_{(i,j)\in S} \xi_{ij}^2 + C_t \sum_{(i,j)\in T} \xi_{ij}^2$$

subject to

$$w^T\left(s(\pi_i) - s(\pi_j)\right) \geq 1 - \xi_{ij},\ \ \xi_{ij} \geq 0,\ \ (i,j) \in S. \quad (6)$$

$$\left|w^T\left(s(\pi_i) - s(\pi_j)\right)\right| \leq 1 + \xi_{ij},\ \ \xi_{ij} \geq 0,\ \ (i,j) \in T. \quad (7)$$

$$w_{kl} \geq w_{k,l+1},\ \ k = 1,\ldots,K,\ \ l = 1,\ldots,L-1. \quad (8)$$

The new constraints Equation (7) enforce the learned weight vector w should give similar grades to pairs with the same degree of relevance. The parameter C_t controls the contribution of tied pairs in the learning process. The optimization problem is still a convex quadratic programming and thus can be solved efficiently with off-the-shelf solvers.

We measure the performance over tied pairs by the averaged difference assigned by the learned weight vector:

$$\text{diff} = \frac{1}{|T|} \sum_{(i,j)\in T} \left|w^T\left(s(\pi_i) - s(\pi_j)\right)\right|.$$

We change the value of C_t from 0 to 1 and show the performance measured by both prediction accuracy on test set and the average score difference defined above in Figure 20. It can be observed that when the value of C_t grows, the averaged score difference reduce significantly. Therefore, the tied pairs are better captured by the learned metric. Moreover, the accuracy on test set slightly improves, which implies that introducing tied pairs can help us to improve the modeling the preference of users.

5.5 Active Learning

We have shown that the selection of the ranking pairs used for training has a significant impact on the performance of DCGLearn. A natural question is how we can select ranking pairs that are more effective for the parameter learning process. To this end, we investigate how to explore active learning methods to select more effective ranking pairs. In particular, we apply the SimpleMargin method for active learning for SVM proposed in [23]. Intuitively, the SimpleMargin method chooses the ranking pairs with lowest confident under the current estimation. In our formulation, assume that we have a pool $P = \{(\pi_i, \pi_j)\}$ of unlabeled ranking pairs, i.e., pairs without preference label. At each iteration, we choose the pair (π_i, π_j) of rankings from the pool such that

$$(\pi_i, \pi_j) = \underset{(\pi_i,\pi_j)\in P}{\text{argmin}} \left|w^T\left(s(\pi_i) - s(\pi_j)\right)\right|.$$

Then, we obtain the preference label the selected pair, add it into the training data and update the model. We report the performance on simulation data sets Data 1 and Data 2 with respect to the number of labeled pairs in Figure 21 and Figure 22, respectively. We observe from the figures that active learning method obtains better prediction accuracy with less labeled training data compared with training sets with randomly selected training ranking pairs. Thus, we conclude that active learning can be used to select ranking pairs that are more effective to the learning process and thus can be used to improve the prediction accuracy and reduce the effort of human labeling of preference data.

We also apply the SimpleMargin algorithm to Web10K and Web30K data set. In Figure 23(a) and Figure 23(b), we report the prediction accuracy with respect to the number of labeled training pairs. We can see that the SimpleMargin algorithm outperforms the supervised learning method in most cases, indicating that active learning can select effective pairs and improve the accuracy. We also notice that in Figure 23(a), when the number of labeled training examples is very small, active learning is not as good as random selection. We suspect that this is because in active learning the training examples are selected according to current model estimation. However, in our case, the model may not be very accurate as a result of the small number of existing training pairs, which makes it difficult to select effective pairs.

In Figure 23(c), we observe similar results on the News data set. The active learning method needs much less training data to learning a good evaluation metric. In this case, we can also observe that the learning curve is not quite smooth. We suspect that this is caused by the noise in the training data.

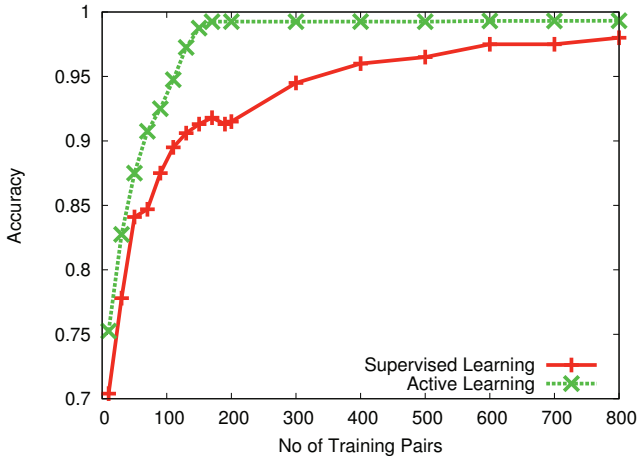


Fig. 21. Prediction accuracy of active learning vs. supervised learning with respect to the number of training pairs on Data 1

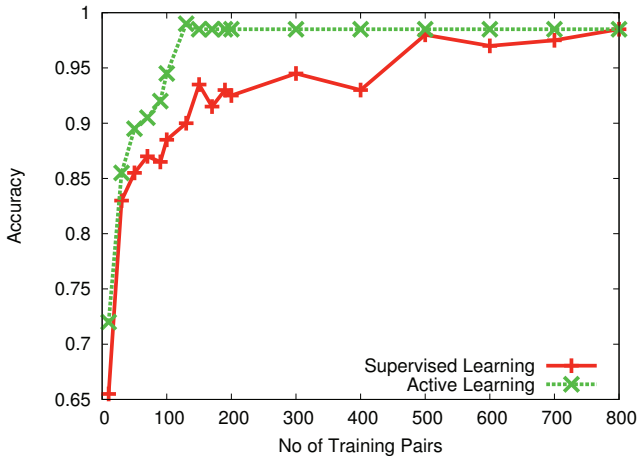
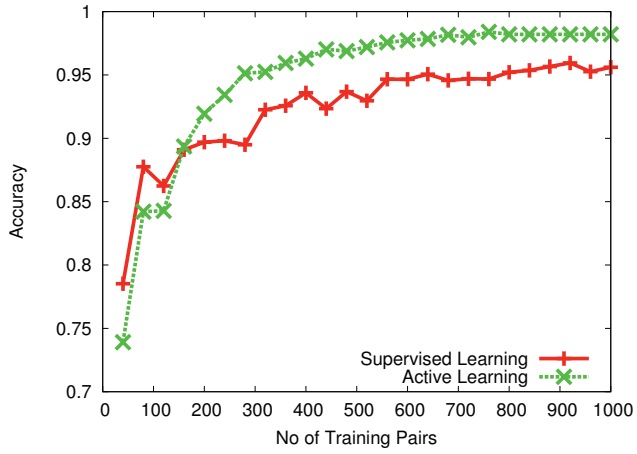


Fig. 22. Prediction accuracy of active learning vs. supervised learning with respect to the number of training pairs on Data 2

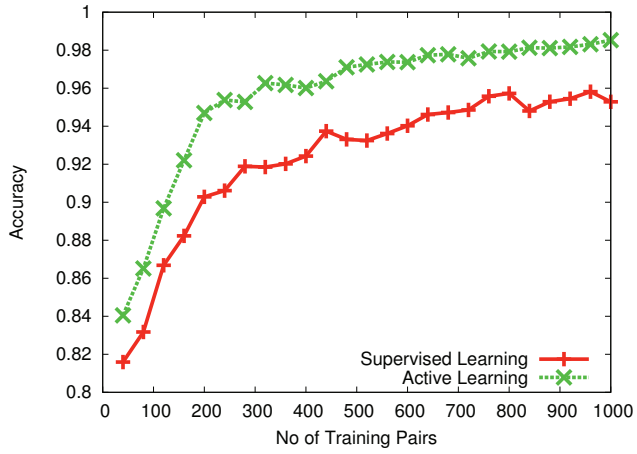
6 Conclusions and Future Work

In this paper, we investigate the important issue of coherency of DCG as an evaluation metric for Web search. Our analytic and empirical analyses show that the DCG is incoherent in general, i.e., different gain vectors can lead to different conclusions about the performance of a ranking function in the context of multi-grade relevance judgments. Therefore, it is very important to select the right parameters for DCG in order to obtain meaningful comparisons of ranking functions. To address this problem, we propose to learn the DCG gain values and discount factors from users' preference judgements of rank lists. In particular, we develop a model to learn the parameters of DCG as a linear utility function and formulate the problem as a convex quadratic programming problem. Results on simulations as well as real-world data suggest the effectiveness of the proposed methods.

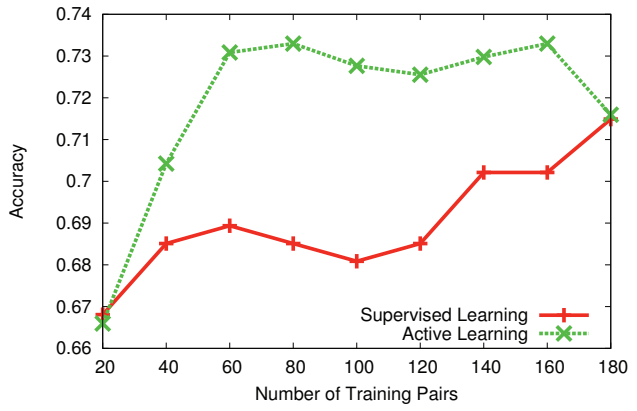
For future research, we plan to explore the learning framework according to different query types and possibly preferences at much finer granularity of personal preference of users; we also plan to investigate how to use user feedback data to generate paired preferences for training, and how to generalize DCG to nonlinear utility functions to model more sophisticated requirements of evaluation metrics, such as diversity and recency. Moreover, learning the parameters for other evaluation metrics such as NDCG is an important



(a) Web10K



(b) Web30K



(c) News Search

Fig. 23. Prediction accuracy of active learning vs. supervised learning on Web10K, Web30K and news search data sets

direction to investigate.

ACKNOWLEDGEMENTS

Part of the work is supported by NSF IIS-1116886, NSF IIS-1049694 and Yahoo! Faculty Research and Engagement award.

REFERENCES

[1] K. Zhou, H. Zha, G.-R. Xue, and Y. Yu, "Learning the gain values and discount factors of dcg," in *Proceedings of Beyond Binary Relevance Workshop, SIGIR2008*, 2008.

[2] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 391–398.

[3] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 271–278.

[4] T. Joachims, "A support vector method for multivariate performance measures," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 377–384.

[5] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya, "Structured learning for non-smooth ranking losses," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 88–96.

[6] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[7] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000, pp. 41–48.

[8] J. Kekäläinen, "Binary and graded relevance in IR evaluations - Comparison of the effects on ranking of IR systems," *Information Processing and Management*, vol. 41, pp. 1019–1033, 2005.

[9] E. M. Voorhees, "Evaluation by highly relevant documents," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 74–82.

[10] T. Sakai, "On the reliability of information retrieval metrics based on graded relevance," *Information Processing and Management*, vol. 43, no. 2, pp. 531–548, 2007.

[11] C. Buckley and E. M. Voorhees, "Evaluating evaluation measure stability," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000, pp. 33–40.

[12] E. Kanoulas and J. A. Aslam, "Empirical justification of the gain and discount function for ndcg," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*. New York, NY, USA: ACM, November 2009, pp. 611–620.

[13] F. Radlinski and N. Craswell, "Comparing the sensitivity of information retrieval metrics," in *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10*. New York, New York, USA: ACM Press, 2010, p. 667.

[14] M. Sanderson, M. L. Paramita, P. Clough, and E. Kanoulas, "Do user preferences and evaluation measures line up?" in *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10*, ser. SIGIR '10. New York, New York, USA: ACM Press, 2010, p. 555.

[15] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, "Novelty and diversity in information retrieval evaluation," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 659–666.

[16] T. Sakai and R. Song, "Evaluating diversified search results using per-intent graded relevance," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*. New York, New York, USA: ACM Press, 2011, p. 1043.

[17] J. J. Louviere, D. A. Hensher, and J. D. Swait, *Stated choice methods: analysis and application*. Cambridge University Press, 2000.

[18] J. D. Carroll and P. E. Green, "Psychometric methods in marketing research: Part II, Multidimensional scaling," *Journal of Marketing Research*, vol. 34, 1997.

[19] O. Chapelle and Z. Harchaoui, "A machine learning approach to conjoint analysis," in *Advances in Neural Information Processing Systems*, Y. W. Saul, L.K. and L. Bottou, Eds., vol. 17. MIT Press, 2005, pp. 257–264.

[20] T. Evgeniou, C. Boussios, and G. Zacharia, "Generalized robust conjoint estimation," *Marketing Science*, vol. 24, no. 3, pp. 415–429, 2005.

[21] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 133–142.

[22] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *J. Mach. Learn. Res.*, vol. 5, pp. 1391–1415, December 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1044706

[23] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *JMLR*, vol. 2, pp. 45–66, March 2002. [Online]. Available: http://dx.doi.org/10.1162/153244302760185243

Ke Zhou received MS degree in computer science and engineering from Shanghai Jiao-Tong University in 2010. He is pursuing his Ph.D. degree at College of Computing, Georgia Institute of Technology. His research interests include information retrieval, machine learning and web image mining.

Hongyuan Zha received the B.S. degree in mathematics from Fudan University in Shanghai in 1984, and the Ph.D. degree in scientific computing from Stanford University in 1993. He was a faculty member of the Department of Computer Science and Engineering at Pennsylvania State University from 1992 to 2006, and he worked from 1999 to 2001 at Inktomi Corporation. His current research interests include computational mathematics, machine learning applications and information retrieval.

Yi Chang joined Yahoo! in 2006. He is leading the ranking science team to work on multiple vertical search and relevance ranking projects. His research interest includes Information Retrieval, Social Computing, Machine Learning, Data Mining and Natural Language Processing.

Gui-Rong Xue received his PhD degree from Shanghai Jiaotong University in 2006. He is a faculty member of Department of Computer Science and Engineering at Shanghai Jiao-Tong University. His research interests are machine learning, data mining and information retrieval.