Learning to Re-rank Web Search Results with Multiple Pairwise Features

Changsung Kang[†], Xuanhui Wang[†], Jiang Chen[‡], Ciya Liao[§], Yi Chang[†], Belle Tseng[†], Zhaohui Zheng[†]
[†]Yahoo! Labs, Sunnyvale, CA 94089
[‡]Google, Mountain View, CA 94043
[§]Microsoft Bing, Mountain View, CA 94043
{ckang,xhwang,yichang,belle,zhaohui}@yahoo-inc.com, criver@gmail.com, cliao@microsoft.com

ABSTRACT

Web search ranking functions are typically learned to rank search results based on features of individual documents, i.e., pointwise features. Hence, the rich relationships among documents, which contain multiple types of useful information, are either totally ignored or just explored very limitedly. In this paper, we propose to explore *multiple* pairwise relationships between documents in a learning setting to *re-rank* search results. In particular, we use a set of *pairwise features* to capture various kinds of pairwise relationships and design two machine learned re-ranking methods to effectively combine these features with a base ranking function: a pairwise comparison method and a pairwise function decomposition method. Furthermore, we propose several schemes to estimate the potential gains of our re-ranking methods on each query and selectively apply them to queries with high confidence. Our experiments on a large scale commercial search engine editorial data set show that considering multiple pairwise relationships is quite beneficial and our proposed methods can achieve significant gain over methods which only consider pointwise features or a single type of pairwise relationship.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Design, Experimentation

Keywords

Pairwise features, Re-rank

WSDM'11, February 9–12, 2011, Hong Kong, China.

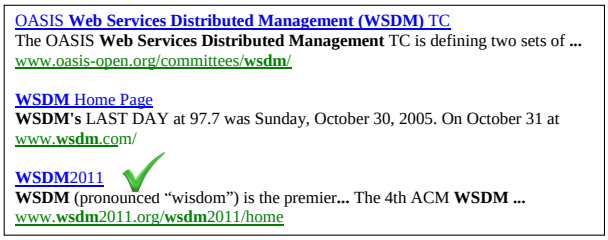


Figure 1: Search results for query "WSDM"

1. INTRODUCTION

Commercial web search engines rely on fine-tuned ranking functions to select search results and thus designing good functions is critical for the success of search engines. Recently, learning to rank techniques have been studied extensively and different learning methods have been proposed [3, 4, 5, 23, 25, 27]. An effective ranking function does not only depend on appropriate learning algorithms but also informative features.

The most commonly used features are pointwise features, i.e., a set of features defined for individual documents. This type of features can be computed easily and a ranking function trained on these features is reasonably effective. However, such a ranking function can be still coarse due to the limited information captured by pointwise features. For example, in Figure 1, a ranking function trained on pointwise features can return a relevant result for query "WSDM", but it is in the third position. To bring this document to the top position, more rich features such as pairwise click preference features in a certain time period can be considered.

Indeed, many research works have been proposed to leverage additional resources for learning to rank, notably click logs. A common strategy is to estimate useful signals from click logs [19] and use them as surrogate relevance labels [18, 7, 9, 11] or to define, usually pointwise, ranking features [1, 17, 6]. Besides click logs, there are other data sources which contain different types of pairwise relationships such as parent-child hyperlink relationship and inter-document similarity relationship. Even for a single data source such as click logs, multiple pairwise preference relationships can be extracted. These pairwise relationships (e.g., SkipAbove) have been shown to be more reliable than pointwise ones (e.g., click counts) [19] and thus provide valuable information for learning to rank. Unfortunately, few work has fully explored

them in a unified way. For example, only click logs were used to derive surrogate relevance labels in [18]. [22] and [21] explored document pairwise relationship to regularize the document ranking scores. [24] and [15] used the concatenated two document feature vectors as pairwise features. One limitation of these works is that only a single type of pairwise relationship is experimented with.

In this paper, we explore multiple pairwise relationships among documents for learning to rank. However, there are several challenges that need to be addressed. (1) Pairwise relationship can be affinitive (e.g., similar documents) or discriminative (e.g., SkipAbove) and each pairwise relationship can be noisy in a different way. How to combine multiple relationships together to complement each other is nontrivial. (2) Given n documents, the pairwise information has an order of $O(n^2)$ space complexity. It is impractical to handle such large information in the web scale.

To tackle the first challenge, we use *pairwise feature* vectors to capture various kinds of pairwise relationships, where each entry in a pairwise feature vector can correspond to a pairwise relationship, and use learning algorithms to optimally combine them. Note this is different from most of existing learning to rank work such as [18] where discriminative pairwise relationships are used as surrogate of editorial labels, instead of features. To tackle the second challenge, we utilize a re-ranking strategy by training a second ranking function using the pairwise features to re-rank the top results of a base ranking function. We specifically propose two machine learned re-ranking methods: a pairwise comparison method and a pairwise function decomposition method. Furthermore, we propose several schemes to estimate the potential gains of our re-ranking methods on each query and selectively apply them to queries with high confidence. Our experiments on a large scale commercial search engine editorial data set show that considering multiple pairwise relationships is quite beneficial and we can achieve significant gain over methods which only consider pointwise features or a single type of pairwise relationship.

The rest of the paper is organized as follows. In Section 2, we introduce related work. We define our problem in Section 3 and describe our re-ranking algorithms in Section 4. In Section 5, we describe our schemes to estimate the potential gain for each query. We present our experimental results in Section 6 and conclude our paper in Section 7.

2. RELATED WORK

In recent years, the web search ranking is usually formulated as a supervised machine learning problem, i.e., learning to rank. These approaches are capable of combining different kinds of features to train ranking functions. The existing methods can be categorized into local ranking and global ranking.

2.1 Local Ranking

The category of local ranking has been studied for a while [18, 4, 12, 25, 8, 26, 14]. In this category, only pointwise features are considered and the learning is to optimize towards the pairwise preference labels, usually formulated as regression or classification problems. For example, RankSVM [18] uses support vector machines to learn a ranking function from preference data. RankNet [4] applies neural network and gradient descent to obtain a ranking function. RankBoost [12] applies the idea of boosting to construct an efficient ranking function from a set of weak ranking functions. The studies reported in [26] proposed a framework called GBRank using gradient descent in function spaces. Since no pairwise feature is considered, this type of methods are usually efficient but the learning capability is limited.

2.2 Global Ranking

Our re-ranking frameworks are more related to the category of global ranking. In global ranking, a ranking model takes all the documents as input and predict their ranking scores jointly. In general, a ranking model F is in the form of

$$\mathbf{y} = F(\mathbf{X})$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ represents the feature vectors corresponding to all the n documents in consideration and $\mathbf{y} = \{y_1, \ldots, y_n\}$ represents the ranking scores assigned to the documents. Some work such as [5] defines the loss function in listwise based on pointwise features. Other existing global ranking algorithms exploit relationships among objects [6, 10, 15, 17, 21, 22, 24] but in a very limited way. Typically, these approaches rely on a single type of relationship among objects or the simple concatenation of pointwise features.

Specifically, [6] uses user clicks to re-rank top search results based on Click Chain Model (CCM), but only click information is considered. [10] only explores the inter-document similarity to regularize initial ranking scores to improve the relevance of ranking. Recently, ranking based on continuous conditional random fields [21] and ranking relational objects [22] are also proposed to explore the inter-document relationship to regularize ranking scores but formulated in a learning framework. Their ranking models are in the form of

$$\mathbf{y} = F(H(\mathbf{X}; \omega), \mathbf{R}) \quad (1)$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ represents feature vectors and $\mathbf{R}$ denotes an $n \times n$ matrix representing relationships among the n objects. Then, F is defined to be a solution of the following minimization problem.

$$F(H(\mathbf{X}; \omega), \mathbf{R}) = \arg\min_{\mathbf{z}} \{l_1(H(\mathbf{X}; \omega), \mathbf{z}) + \beta l_2(\mathbf{R}, \mathbf{z})\} \quad (2)$$

where $\mathbf{z}$ denotes a vector of any possible ranking scores for the documents. The first objective l_1 measures the difference between H and $\mathbf{z}$ and the second objective l_2 measures the inconsistency between elements in $\mathbf{z}$ under the relationship matrix $\mathbf{R}$. β is a non-negative coefficient that controls the trade-off between the two objectives. Solving the minimization problem corresponds to finding the best parameters ω. The first objective $l_1(H(\mathbf{X}; \omega), \mathbf{z})$ is simply $||H(\mathbf{X}; \omega) - \mathbf{z}||^2$ where $||.||$ denotes L_2 norm. However, the second objective $l_2(\mathbf{R}, \mathbf{z})$ is defined differently for each ranking task (pseudo relevance feedback or topic distillation) in their papers. Although in principle it is possible to explore different relationships in their framework, careful design of different l_2 functions for each relationship is needed to differentiate different relationships. If the same l_2 is used, it is equivalent to collapse several relationships into a single one. In contrast, in our model, various types of pairwise relationships among documents can be leveraged by representing them as pairwise features and this has not been explored in the above models.

Our specific methods are closely related to Ranking by Pairwise Comparison (RPC) [15] and BoltzRank [24]. RPC

is similar to our pairwise comparison method (Section 4.1) since they both perform two steps for ranking: (i) they first learn pairwise preferences and (ii) they combine the pairwise preferences into a ranking. Also, BoltzRank is similar to our pairwise function decomposition method (Section 4.2) since they both decompose a ranking model into two parts: (i) individual potential and (ii) pairwise potential. We adapt these methods for the re-ranking problem. The major difference is our use of a base ranking function. Compared to RPC, our pairwise comparison method performs a local search to minimize an objective function, which is made feasible by leveraging a base ranking function. In our pairwise function decomposition method, we use a base ranking function as individual potential. The number of training instances for BoltzRank is $O(Qn^2)$ where Q is the number of queries and n is the number of documents for each query. Thus, we may not be able to use a huge number of queries in the training data, which is not desirable for the generalization ability of ranking models. On the other hand, we can use much more queries in the training data for a base ranking function. Hence, we can add more robustness to our models by incorporating a base ranking function. More importantly, our work uses multiple much more meaningful pairwise features (Section 3.2) while RPC and BoltzRank rely on the simple concatenation of pointwise features as the single pairwise feature.

3. PROBLEM FORMULATION

We formally define our problem in this section. We first review the learning to rank based on pointwise features. Then we introduce our pairwise relationships, define features upon these relationships, and describe our problem of learning to re-rank.

3.1 Learning to Rank on Pointwise Features

Conventional learning to rank depends on pointwise features, i.e., a set of features defined for individual documents. Given a query q, let $\mathcal{D}_q = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ denote the feature vectors of all the documents to be ranked, where each $\mathbf{x}_i \in \mathbb{R}^d$ has d dimensions. For simplicity, we omit a notation of q and use $\mathcal{D}$ to represent $\mathcal{D}_q$ when it is clear from the context. In a ranking problem, $\mathcal{D}$ is given as input and a permutation τ of $\{1, \ldots, n\}$ is returned as output. $\mathbf{x}_i$ is ranked higher than $\mathbf{x}_j$ if $\tau(\mathbf{x}_i) < \tau(\mathbf{x}_j)$ and this means $\mathbf{x}_i$ is more relevant to q than $\mathbf{x}_j$. In a typical web search ranking problem, a *ranking function* $f : \mathbb{R}^d \to \mathbb{R}$ is typically trained and applied to $\mathcal{D}$. A permutation or ranking τ is generated by ordering the $f(\mathbf{x})$ in the descending order.

In most existing work, $\mathbf{x}$ only consists of pointwise features such as the TF-IDF matching score between the query and the document. We use b to denote such a base ranking function. A ranking function trained on these features is reasonably effective. However, pointwise features might not be good enough since the web search problem is admitted to be very complex. On the other hand, there are rich pairwise relationships between documents. In the next section, we define our pairwise features based on the pairwise relationships.

3.2 Pairwise Features

In this section, we present several types of pairwise relationships between documents and then define our pairwise features to capture these relationships.

3.2.1 Pairwise Click Preference

Click logs represent an important source of users' relevance feedback and have been used for estimating document relevance [19] or deriving pointwise features [17]. We are interested in using click logs to discover relative preference information. As shown in [19], click based relative preference is more accurate than absolute preference. However, relative preference is still too noisy to be used as training labels. Instead, we use them as features. Given u_i and u_j, a discriminative feature should be able to differentiate their relevance to a query with high confidence. To this end, we define and collect the following pairwise click features:

- cc_{ij}: the number of sessions in which both u_i and u_j were clicked
- cnc_{ij}: the number of sessions in which u_i was clicked but u_j was not clicked
- ncc_{ij}: the number of sessions in which u_i was not clicked but u_j was clicked
- t_i: the average dwell time on u_i
- t_j: the average dwell time on u_j

where a session is defined for a unique (user, query) pair. The session starts when a user issues a query and ends after a certain idle time on the user side or the user issues a different query.

While each individual features can be noisy, the combination of these features provides a more direct and reliable relevance comparison of two documents. Note that we include dwell time information of documents to deal with noisy clicks. Clicks are more correlated to *perceived relevance* on search results pages than the true document relevance: many clicks are due to attractive presentation or the high position of documents on the results page. Some unrelevant documents may be clicked but then users leave shortly. The dwell time information can be used to calibrate these noisy clicks.

3.2.2 Document Similarity

The similarity between two documents has been extensively studied in information retrieval. It has recently been used for adjusting base ranking scores [10]. We assume that documents that are similar to relevant documents are likely to be relevant, due to the clustering hypothesis [16]. Under this assumption, we may boost some documents based on the document-document similarity after a base ranking is decided. However, in contrast to [10] which use the document similarity as a single re-ranking signal, we define several pairwise features based on document similarity and further combine them with other re-ranking features in our models. The following is some examples of similarity features:

- $sim(u_i, u_j)$: the similarity between u_i and u_j
- $sim(u, +)$: the similarity between u and the document(s) that received the most positive feedback in click logs
- $sim(u, -)$: the similarity between u and the document(s) that received the most negative feedback in click logs

$sim(u,+)$ and $sim(u,-)$ can be easily computed from click logs. For example, a document with a negative feedback can be identified by *SkipAbove* information [20]: for two documents u_i and u_j where u_i is ranked higher than u_j, but u_i is not clicked while u_j is clicked. we regard that u_i received negative feedback.

3.2.3 Parent-child Relationship

It is common that a web page and some of its child pages appear together in a search results page. Let u_i and u_j be two web pages under the same website. u_i is said to be a parent of u_j if the url of u_i is a prefix of that of u_j. Symmetrically, we call u_j a child of u_i. In general, search engines tend to rank parent pages higher than their child pages. However, this bias is incorrect when a parent page is too general for a given query while a child page matches the query better. Consider a query "carmax used". Most commercial search engines rank the parent page www.carmax.com before the child page www.carmax.com/enUS/car-search/used-cars.html in the first results page. It is clear that this ranking is not optimal since the user is intended to find "used" cars in the child page. We can use the simple ternary feature

$$pc_{ij} = \begin{cases} 1 & \text{if } u_i \text{ is a parent of } u_j, \\ -1 & \text{if } u_i \text{ is a child of } u_j, \\ 0 & \text{otherwise} \end{cases}$$

to represent the parent-child relationship.

3.2.4 Concatenated Pointwise Features

Let $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]$ be a pointwise feature vector of document u_i. Then, we can define a simple pairwise feature vector between u_i and u_j by literally concatenate the two pointwise feature vectors:

$$\mathbf{x}_{ij} = [\mathbf{x}_i, \mathbf{x}_j]$$

Actually, this is the only pairwise feature used in [15]. We also tried to define the difference or ratio between two pointwise feature vectors as a new pairwise feature. However, in our feature selection process, we have observed that this type of feature is not very useful.

3.3 Learning to Re-Rank on Pairwise Features

Most of the above features (except for the features in Section 3.2.4) are inherently pairwise and they are very difficult, if ever possible, to be derived from pointwise features. Obviously, these true pairwise features can be hardly used for a base ranking function since they are defined in terms of document pairs.

Let $\mathcal{P}_q = \{\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j} \mid \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_q, \mathbf{x}_i \neq \mathbf{x}_j\}$ be the set of pairwise feature vectors for a query q where $\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j} \in \mathbb{R}^{d_p}$ concatenates all the features defined in Section 3.2 between document $\mathbf{x}_i$ and $\mathbf{x}_j$. A direct way of utilizing these features is to concatenate the pairwise feature vectors with the pointwise feature vectors together and then train a giant ranking function. This is obviously impractical in both training and ranking. We thus propose a re-ranking strategy to combine these pairwise features with a base ranking function trained on pointwise features.

Formally, given a ranking list τ_b of top n results obtained from the base ranking function b, our goal is to learn a re-ranking policy r that produces a new ranking τ_r by considering pairwise features $\mathcal{P} = \{\mathcal{P}_q \mid q \text{ is a training query}\}$.

Note that the above formulation of re-ranking is more general than global ranking described in Section 2.2 and any types of pairwise relationships between documents can be used in this framework. Furthermore, in our re-ranking problem, it is not necessary to assign a new ranking score for each document and then produce τ_r. In the following, we present two algorithms: the pairwise comparison method, which directly generates the permutation, and the pairwise function decomposition method, which first generates new ranking scores and then derives a new permutation.

4. LEARNING TO RE-RANK ALGORITHMS

In this section, we present our learning to re-rank approaches based on pairwise features between documents.

4.1 Pairwise Comparison Method

In this section, we introduce a straightforward approach which learns the relative relevance for document pairs and then re-ranks results to minimize the number of discordant pairs.

4.1.1 Learning Pairwise Preferences

The goal of this section is to learn a function that predicts the probability that $\mathbf{x}_i$ is more relevant than $\mathbf{x}_j$ to a query. Given all the pairwise features, we have the following training data for each training query:

$$\mathcal{T}_q = \{\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}, \text{pref}(l_i, l_j) \mid i, j \in \{1, \ldots, N\}, i \neq j\}$$

where l_i is a numerical label given by human editors to $\mathbf{x}_i$ out of a finite set of labels L (e.g., $L = \{4, 3, 2, 1, 0\}$) and $\text{pref}(l_i, l_j)$ is interpreted as the probability $p(\mathbf{x}_i \succ \mathbf{x}_j)$ that $\mathbf{x}_i$ is more relevant than $\mathbf{x}_j$. A simple way to set $\text{pref}(l_i, l_j)$ is

$$\text{pref}(l_i, l_j) = \begin{cases} 1 & l_i > l_j \\ 0.5 & l_i = l_j \\ 0 & l_i < l_j. \end{cases}$$

Then, we could solve a regression problem with the training data and interpret the response of the prediction as a probability. However, this method ignores the difference between l_i and l_j. Alternatively, we may compute $\text{pref}(l_i, l_j)$ based on the difference $l_i - l_j$. By doing so, we are assuming that there is some uncertainty in labels l_i and l_j. In a typical process of obtaining labels, human editors are restricted to choose one label out of a small, predefined set. Suppose that an editor gave $\mathbf{x}_i$ the label 3. If the editor were allowed to give a real value (not restricted to the predefined set) to the document, the possible values of the label would form a distribution with the average around 3. Also, the labels are noisy because editors often make incorrect judgments or their judgments can be subjective.

We model the uncertainty of the pointwise labels as follows. We define a random variable R_l for each label $l \in L$ and assume a Gaussian distribution for each R_l:

$$p(R_l) = \mathcal{N}(R_l \mid l, \sigma_l^2)$$

Although we may try to estimate the variance σ_l for each $l \in L$, we make an assumption that we have a common variance σ for all $l \in L$.

Given these random variables, it is straightforward to de-

rive the pairwise preference probability $\text{pref}(l_i, l_j)$:

$$\begin{aligned}\text{pref}(l_i, l_j) &= p(\mathbf{x}_i \succ \mathbf{x}_j) \\ &= p(R_{l_i} > R_{l_j}) \\ &= p(R_{l_i} - R_{l_j} > 0) \\ &= \int_0^\infty \mathcal{N}(R_l \mid l_i - l_j, 2\sigma^2) dR_l\end{aligned}$$

We choose σ such that $\text{pref}(l_{max}, l_{min}) = 1$ where l_{max} is the largest label and l_{min} is the smallest label in L. We then apply the gradient boosting method (GBDT) [13] on our training data $\{\mathcal{T}_q \mid q \text{ is a training query}\}$ to obtain a function $h(\mathbf{w_{xy}})$ which can predict the relative relevance of two documents to a query.

4.1.2 Re-ranking with Pairwise Preferences

We propose an algorithm for re-ranking results using the pairwise preference function $h(\mathbf{w_{xy}})$. Given a ranking list τ_b of top n results obtained from the base ranking function b, a straightforward way of re-ranking results using the pairwise preference function $h(\mathbf{w_{xy}})$ is to swap $\mathbf{x}$ and $\mathbf{y}$ if $\tau_b(\mathbf{x}) > \tau_b(\mathbf{y})$ and $h(\mathbf{w_{xy}}) > \alpha$ where $\alpha > 0.5$ is a parameter that controls the confidence of swapping. A more principled way of re-ranking is to derive an objective function defined in terms of $h(\mathbf{w_{xy}})$ and find a ranking that optimizes the objective function. We minimize the following objective function

$$\tau_r = \arg\min \sum_{\tau_r(\mathbf{x}_i) > \tau_r(\mathbf{x}_j)} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}). \quad (3)$$

It is known that finding the optimal solution that minimizes (3) is NP complete [2]. Hence, we propose a greedy algorithm to minimize (3) in Algorithm 1. We start from a base ranking $\tau_r = \tau_b$. At each step, we generate a candidate ranking τ_r' by swapping two documents in τ_r. We evaluate the objective function value for each candidate ranking and select the best one. If the best ranking improves the current ranking τ_r, we take it to the next step. We repeat this until we cannot improve the ranking further.

The time complexity of Algorithm 1 is $O(mn^4p)$ where m is the number of iterations and p is the time to compute each h. The time complexity can be reduced by the following heuristics: (1) At each iteration, we may sort the list of candidate swaps by h values and try the candidate with highest value first. (2) We can reduce the number of steps required to compute s' in line 9 from $O(n^2p)$ to $O(np)$ using a differential update since τ_r and τ_r' differ only in two positions and we just need to make adjustments for the terms that are affected: all the documents between $\tau_r(\mathbf{x}_i)$ and $\tau_r(\mathbf{x}_j)$. Thus, the overall time complexity (worst-case) is reduced to $O(mn^3p)$. We perform the differential update for s' as follows. To simplify the description, we need to use the inverse τ^{-1} of a permutation τ (note that a permutation τ is a bijection from documents to positions): $\tau^{-1}(i)$ is the document at position i. Assume that $\tau_r(\mathbf{x}_i) < \tau_r(\mathbf{x}_j)$. Then, we have

$$\begin{aligned}s' = s &- \Big\{h(\mathbf{w}_{\tau_r^{-1}(\tau_r(i)+1)\mathbf{x}_i}) + \ldots + h(\mathbf{w}_{\mathbf{x}_j\mathbf{x}_i})\Big\} \\ &- \Big\{h(\mathbf{w}_{\mathbf{x}_j\tau_r^{-1}(\tau_r(i)+1)}) + \ldots + h(\mathbf{w}_{\mathbf{x}_j\tau_r^{-1}(\tau_r(j)-1)})\Big\} \\ &+ \Big\{h(\mathbf{w}_{\tau_r^{-1}(\tau_r(i)+1)\mathbf{x}_j}) + \ldots + h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})\Big\} \\ &+ \Big\{h(\mathbf{w}_{\mathbf{x}_i\tau_r^{-1}(\tau_r(i)+1)}) + \ldots + h(\mathbf{w}_{\mathbf{x}_i\tau_r^{-1}(\tau_r(j)-1)})\Big\}.\end{aligned}$$

Algorithm 1 Greedy algorithm to re-ranking search results with pairwise preferences

Input: $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $b(\mathbf{x})$, $h(\mathbf{w_{xy}})$
Output: A new ranking τ_r of $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$

```
1: Let τ_b be the ranking of {x_1, ..., x_n} given by b.
2: Initialize τ_r = τ_b
3: s = Σ_{τ_r(x_i) > τ_r(x_j)} h(w_{x_i x_j})
4: while True do
5:    improved = False
6:    for i = 1 to n − 1 do
7:       for j = i + 1 to n do
8:          τ_r' = τ_r with i and j swapped
9:          s' = Σ_{τ_r'(x_i) > τ_r'(x_j)} h(w_{x_i x_j})
10:         if s' < s then
11:            s = s', s_i = i, s_j = j
12:            improved = True
13:         end if
14:      end for
15:   end for
16:   if improved then
17:      Update τ_r: swap τ_r(x_{s_i}) and τ_r(x_{s_j}).
18:   else
19:      break
20:   end if
21: end while
22: return τ_r
```

The number of terms added or subtracted is $4\big(\tau_r(j) - \tau_r(i)\big) - 2$. Hence, given the objective function value for τ_r, the computation of s' for τ_r' (line 9) takes $O(np)$ steps instead of $O(n^2p)$ steps.

4.2 Pairwise Function Decomposition Method

The pairwise comparison method may be impractical at query time unless we re-rank a very small number of documents or limit the number of iterations in the greedy search algorithm. Another disadvantage of the pairwise comparison method may be that it does not fully leverage the base ranking function.

In this section, we propose another re-ranking approach, called *pairwise function decomposition method*, which produces a new ranking score for each document by combining the score of a base ranking function and the pairwise score adjustments learnt from pairwise features. This method is more efficient at query time.

4.2.1 Training

We decompose the ranking function into two parts:

$$f(\mathbf{x}) = b(\mathbf{x}) + \sum_{\mathbf{y} \in \mathcal{D}} h(\mathbf{w_{xy}}) \quad (4)$$

where b is the base ranking function and h is a function that effectively enforces relative constraints between pairs of documents. Note that $b(\mathbf{x})$ is a constant value and we only learn h.

For each query, our training data consists of three parts (for easy exposition, we omit the notation of query):

- labels: $\{l_i\}_{i=1}^N$ where l_i is the label of $\mathbf{x}_i$
- scores by the base ranking function: $\{b(\mathbf{x}_i)\}_{i=1}^N$
- pairwise features: $\{\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j} \mid i, j \in \{1, \ldots, N\}, i \neq j\}$

Algorithm 2 Gradient boosting for pairwise function decomposition

Input: Training data: $\{l_i\}_{i=1}^N$, $\{b(\mathbf{x}_i)\}_{i=1}^N$, $\{\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j} \mid i,j \in \{1,\dots,N\}, i \neq j\}$
Output: Gradient boosting trees $h(\mathbf{w}_{\mathbf{xy}})$

1: Initialize $h_0(\mathbf{w}_{\mathbf{xy}}) = \frac{1}{N}\sum_{i=1}^N \big(l_i - b(\mathbf{x}_i)\big)$
2: **for** $k = 1,\dots,M$ **do**
3: **for** each (i,j) such that $i < j$ **do**
4: Compute the negative gradient $\gamma_{i,j}^k = -\Big[\frac{\partial L(h)}{\partial h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})}\Big]_{h(\mathbf{w}_{\mathbf{xy}})=h_{k-1}(\mathbf{w}_{\mathbf{xy}})}$
5: **end for**
6: Fit a regression tree function t_k to $\{\gamma_{i,j}^k\}_{i<j}$
7: Update $h_k(\mathbf{w}_{\mathbf{xy}}) = h_{k-1}(\mathbf{w}_{\mathbf{xy}}) + \eta s_k t_k(\mathbf{w}_{\mathbf{xy}})$ where η is a *shrinkage factor* and s_k is found by the line search to minimize the loss function.
8: **end for**
9: **return** $h_M(\mathbf{w}_{\mathbf{xy}})$

Given the training data, we learn h by solving the following optimization problem.

$$\min_{h\in H}\sum_q \sum_{i\in\{1,\dots,N\}} \frac{1}{2}\Big\{l_i - b(\mathbf{x}_i) - \sum_{j\in\{1,\dots,N\}\setminus\{i\}} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})\Big\}^2$$

We enforce the symmetry of h: $h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})+h(\mathbf{w}_{\mathbf{x}_j\mathbf{x}_i}) = 0$ and achieve this by replacing $h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})$ by $-h(\mathbf{w}_{\mathbf{x}_j\mathbf{x}_i})$ if $i > j$. Then, we can rewrite the loss function as follows.

$$L(h) = \sum_q \sum_{i\in\{1,\dots,N\}} \frac{1}{2}\Big\{l_i - b(\mathbf{x}_i) + \sum_{j<i} h(\mathbf{w}_{\mathbf{x}_j\mathbf{x}_i}) - \sum_{j>i} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})\Big\}^2$$

Similarly, we apply the gradient boosting method (GBDT) [13] to solve this optimization problem. Algorithm 2 summarizes our training procedure. The negative gradient $\gamma_{i,j}^k$ in the algorithm is computed as follows.

$$\begin{aligned}\gamma_{i,j}^k &= -\Big[\frac{\partial L(h)}{\partial h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})}\Big]_{h(\mathbf{w}_{\mathbf{xy}})=h_{k-1}(\mathbf{w}_{\mathbf{xy}})} \\ &= \Big\{l_i - b(\mathbf{x}_i) + \sum_{i'<i} h_{k-1}(\mathbf{w}_{\mathbf{x}_{i'}\mathbf{x}_i}) - \sum_{i'>i} h_{k-1}(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_{i'}})\Big\} \\ &\quad - \Big\{l_j - b(\mathbf{x}_j) + \sum_{j'<j} h_{k-1}(\mathbf{w}_{\mathbf{x}_{j'}\mathbf{x}_j}) - \sum_{j'>j} h_{k-1}(\mathbf{w}_{\mathbf{x}_j\mathbf{x}_{j'}})\Big\}\end{aligned}$$

Note that we may use any type of loss function such as pairwise loss function [27] instead of the regression loss used above and we leave these exploration to the future.

The pairwise function decomposition method can be more robust than the pairwise comparison method described in the previous section. In the pairwise comparison method, the relative relevance between two documents is determined only by the two. In contrast, the pairwise function decomposition method involves all the documents in the result set to adjust the score of each document, which provides more robustness.

Re-ranking at query time using the pairwise function decomposition method is straightforward. The search results given by the base ranking function b are re-ranked by f in (4). The time complexity of re-ranking at query time is $O(n^2p)$ where p is the time to compute each h. Compared to the pairwise comparison method ($O(mn^3p)$), the pairwise function decomposition method is more efficient.

5. QUERIES WITH HIGHLY RE-RANKABLE RESULTS

In this section, we seek to identify queries with highly *re-rankable* search results. Search results are said to be highly re-rankable if they have high potential gains after re-ranking. To estimate potential gains, we can directly rely on some pairwise relationships or the function h learned by the pairwise comparison method or the pairwise function decomposition method. Using these signals, we can control our confidence in re-ranking. This gives us a knob to trade-off high accuracy and low coverage or high coverage and low accuracy.

We propose to use the following four schemes to identify queries for re-ranking with high confidence.

Scheme 1: Re-rank search results for a query if there exist a pair of documents $\mathbf{x}_i$ and $\mathbf{x}_j$ such that $\tau_b(\mathbf{x}_i) > \tau_b(\mathbf{x}_j)$ and $\frac{cnc_{ij}}{ncc_{ij}} > \alpha$ where α is a parameter that controls the confidence of re-ranking.

The condition $\tau_b(\mathbf{x}_i) > \tau_b(\mathbf{x}_j)$ and $\frac{cnc_{ij}}{ncc_{ij}} > \alpha$ means that users skip $\mathbf{x}_j$ and click $\mathbf{x}_i$ in the search results. The higher α, the more likely $\mathbf{x}_i$ and $\mathbf{x}_j$ are swapped in the re-ranked results. However, coverage of re-ranking (number of impacted queries) decreases as α increases. Hence, we can control the trade-off between accuracy and coverage by a single parameter α.

Scheme 2: Re-rank search results for a query if there exist a pair of documents $\mathbf{x}_i$ and $\mathbf{x}_j$ such that $\tau_b(\mathbf{x}_i) > \tau_b(\mathbf{x}_j)$ and $h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}) > \alpha$ where h is the pairwise preference function in the pairwise comparison method or the pairwise potential function in the pairwise function decomposition method.

Since h is trained using various pairwise features including pairwise click features such as cnc_{ij} and ncc_{ij}, it should provide a more robust signal regarding the relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Scheme 3: Re-rank search results for a query if

$$\Big(\sum_{\tau_b(\mathbf{x}_i)>\tau_b(\mathbf{x}_j), h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})>\beta} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})\Big) > \alpha$$

where β is 0.5 for the pairwise comparison method and 0 for the pairwise function decomposition method. Note that $h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}) > 0.5$ for the pairwise comparison method implies that $\mathbf{x}_i$ is more relevant than $\mathbf{x}_j$ and $h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}) > 0$ for the pairwise function decomposition method implies that $\mathbf{x}_i$ is more relevant than $\mathbf{x}_j$. Hence, the value

$$\sum_{\tau_b(\mathbf{x}_i)>\tau_b(\mathbf{x}_j), h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})>\beta} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})$$

is approximately the improvement in the objective function (3) after re-ranking (if all the pairs $\{(\mathbf{x}_i,\mathbf{x}_j) \mid \tau_b(\mathbf{x}_i) > \tau_b(\mathbf{x}_j),\ h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}) > \beta\}$ are swapped). Therefore, we can control the approximate ranking improvement by α.

Scheme 4: Re-rank search results for a query if

$$\Big(\sum_{\tau_b(\mathbf{x}_i)>\tau_b(\mathbf{x}_j)} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j}) - \sum_{\tau_r(\mathbf{x}_i)>\tau_r(\mathbf{x}_j)} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})\Big) > \alpha$$

where τ_r is the ranking after re-ranking. This scheme uses the change in the objective function (3) after re-ranking to

	#Train	#Test	#Feature
base ranking	1.2M	150K	35
re-ranking	5.1M	400K	100
#queries	71K	4K	–

Table 1: Statistics of our data sets

estimate the ranking improvement. In order to compute the value $\sum_{\tau_r(\mathbf{x}_i)>\tau_r(\mathbf{x}_j)} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})$, we do not need to actually trigger re-ranking and change the search results shown to users. Instead, we can first execute re-ranking in memory and compute the value of $\sum_{\tau_r(\mathbf{x}_i)>\tau_r(\mathbf{x}_j)} h(\mathbf{w}_{\mathbf{x}_i\mathbf{x}_j})$. Then, we can decide whether to actually trigger re-ranking based on the above condition.

Although the computation of scheme 3 and 4 may seem expensive, the values of h for all the pairs of documents are used in the re-ranking anyway. Hence, there is no additional cost when re-ranking is triggered.

For all the above four schemes, we can control the trade-off between accuracy and coverage by a single parameter α. How do we determine which query selection scheme and what value of α to use? We may choose a query selection scheme and the value of α based on the minimum level of ranking improvement that we expect. For example, we may want +10% NDCG_5 gain over the base ranking function. Then, the gain-coverage graph (in Section 6) will determine a query selection scheme and the minimum value of α to obtain at least 10% NDCG_5 gain for the queries for which re-ranking is triggered.

6. EXPERIMENTS

In this section, we evaluate our two re-ranking methods based on pairwise features. The objectives of our experiments are: (1) to evaluate the improvement of search result accuracy by the proposed re-ranking methods, (2) to examine the effect of different query selection schemes on the accuracy and the coverage of the re-ranking methods, and (3) to compare the usefulness of different classes of features used in our models.

6.1 Experiment Design

6.1.1 Data Sets

The data sets we use are from a commercial search engine and Table 1 summarizes their statistics. To train a base ranking function, we use a set of training data $(\mathbf{x}, l)$ where a feature vector $\mathbf{x}$ corresponds to a (query, document) pair and l is the label given to $\mathbf{x}$ using five values, $\{4, 3, 2, 1, 0\}$, representing five levels of relevance: perfect, excellent, good, fair, and bad. A feature vector $\mathbf{x}$ contains query-dependent features, document-dependent features and (query, document)-dependent features. We use the top 35 pointwise features (including some text-matching features and some click-related features) currently used by the commercial search engine. In this pointwise training data, there are 1.2M feature vectors.

To train the pairwise comparison method and the pairwise function decomposition method, we need (in addition to the pointwise training data) a set of pairwise feature vectors $\mathbf{w}_{\mathbf{xy}}$ and base ranking scores for all $\mathbf{x}$ and thus the pairwise training data is obtained after we train a base ranking function. For each query, we generate pairwise feature vectors for pairs of documents (only among top 10 documents in the base ranking) for that query. We obtain pairwise click preference features described in Section 3.2.1 from the click logs of the same search engine. In total, we have 100 pairwise features: 30 features for pairwise click preference, document similarity and parent-child relationship and 70 features for the concatenated pointwise features (35 for each document in a pair). Note that these pairwise features cannot be used by a base ranking function, which is a "local" ranking model. In total, there are 5.1M pairwise feature vectors in our data.

The test data is similarly generated. We have 150K pointwise feature vectors and 400K pairwise feature vectors in the test data. In total, we have 71K queries in the training set and 4K queries in the test set.

6.1.2 Algorithms

For the queries in our test data, we compare their rankings given by

- **Base Ranking Function**: gradient boosting trees model [13] trained with the pointwise training data.
- **Pairwise Click-based Swap (PCSwap)**: simple re-ranking algorithm using pairwise click features described in Section 3.2.1, which is similar to the re-ranking algorithm proposed in [6]. Two documents $\mathbf{x}_i$ and $\mathbf{x}_j$ are swapped if $\mathbf{x}_i$ is ranked lower than $\mathbf{x}_j$ and $\frac{cnc_{ij}}{ncc_{ij}} > \alpha$ and $\frac{t_i}{t_j} > \beta$.
- **Pairwise Comparison (PC)**: described in Section 4.1 with different query selection parameters α.
- **Pairwise Function Decomposition (PFD)**: described in Section 4.2 with different query selection parameters α.

The base ranking function serves as a strong baseline which is trained with many well-tuned features used in a commercial search engine. PCSwap is another baseline similar to a recently proposed re-ranking algorithm [6] and represents the state-of-the-art algorithm which explores click logs.

6.1.3 Evaluation Metrics

The evaluation is based on three metrics NDCG_5, NDCG_1 and the pair accuracy. NDCG_k is defined to be

$$\text{NDCG}_k = \frac{1}{Z_k} \sum_{i=1}^{k} \frac{G_i}{\log_2(i+1)}$$

where G_i denotes the label of the document at position i and Z_k represents a normalization factor to guarantee that the NDCG_k for the perfect ranking (among the permutations of the retrieved documents) is 1.

The pair accuracy is the ratio of correct pairs

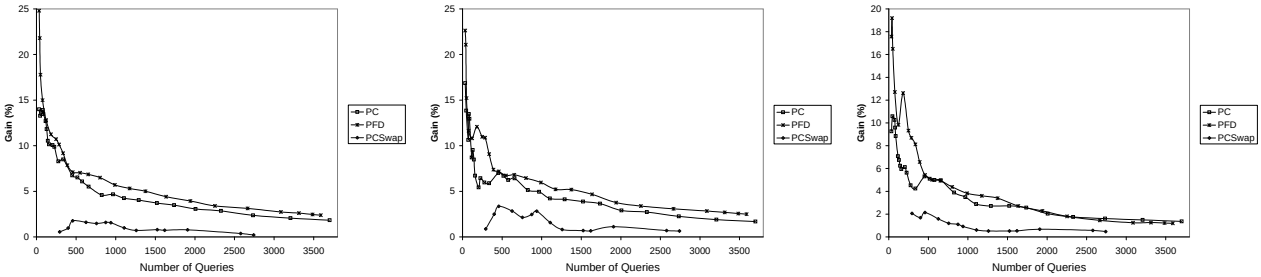
$$\frac{\{(\mathbf{x}_i, \mathbf{x}_j) \mid \tau(\mathbf{x}_i) < \tau(\mathbf{x}_j), l_i > l_j\}}{\{(\mathbf{x}_i, \mathbf{x}_j) \mid \tau(\mathbf{x}_i) < \tau(\mathbf{x}_j)\}}.$$

A metric is computed for each query and the average values over all the queries in our test data are reported.

6.2 Results

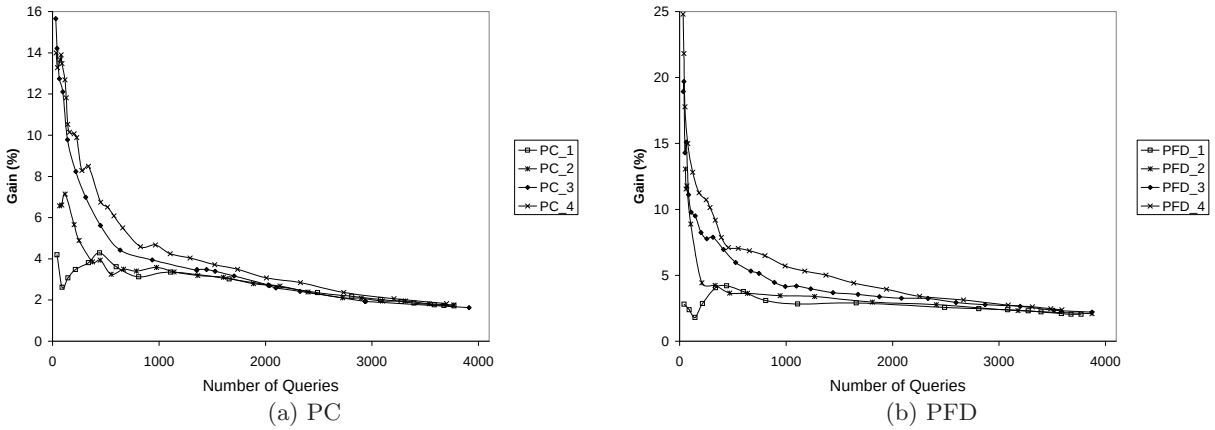
Table 2: Relevance improvements with no query selection scheme. Statistically significant gains ($p \leq 0.001$) are highlighted in bold. Please note that our 2% NDCG gain is much larger than that reported in [6].

	$NDCG_5$	$NDCG_5$ Gain (%)	$NDCG_1$	$NDCG_1$ Gain (%)	Pair Accuracy	Pair Accuracy Gain (%)
Base	0.7434	0	0.7898	0	0.8447	0
PCSwap	0.7464	0.4035	0.7953	0.6964	0.8479	0.3788
PC	0.7545	**1.4894**	0.8015	**1.4769**	0.8542	**1.1200**
PFD	0.7585	**2.0281**	0.8061	**2.0662**	0.8535	**1.0431**



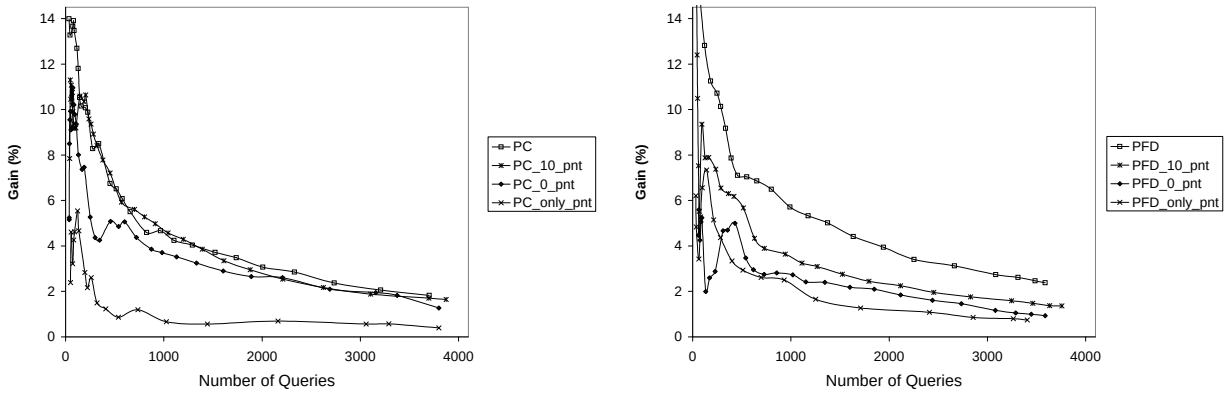
(a) $NDCG_5$ (b) $NDCG_1$ (c) Pair Accuracy

Figure 2: Improvements for affected queries by PC and PFD with different α values. Scheme 4 is used as the query selection scheme. PCSwap is also compared using different α and β values. Gains are against the base ranking function. The graphs show the trade-off between the relevance gain and the query coverage of re-ranking.



(a) PC (b) PFD

Figure 3: Comparing query selection schemes. Scheme i is denoted as PC_i and PFD_i.



(a) PC (b) PFD

Figure 4: Evaluation of features. $NDCG_5$ is compared for different features used in the training. PC and PFD use all features. PC_10_pnt and PFD_10_pnt use all pairwise features and 10 pointwise features. PC_0_pnt and PFD_0_pnt use all pairwise features and no pointwise features. PC_only_pnt and PFD_only_pnt use no pairwise features and all pointwise features. Scheme 4 is used as a query selection scheme.

6.2.1 Relevance Improvement Comparison

We first compare all the algorithms in terms of relevance improvement. Table 2 shows the results of all the 4 algorithms for all the 3 metrics. Note that we do not use any query selection scheme described in Section 5. In this table, we also show the absolute gains of our re-ranking algorithms against the base ranking function. Our proposed methods PC and PFD outperform the baseline statistically significantly. For example, PFD achieves +2.0% $NDCG_5$ gain (0.015 absolute gain), which is a significant improvement considering that our base ranking function is comparable to a commercial search engine function. Compared with our methods, PCSwap can improve over the baseline but the improvement is much smaller. In fact, the best absolute NDCG gain reported in [6] (see their Section 5) is about 0.007 and this is much smaller than our 0.015 absolute gain. Furthermore, our methods are statistically significantly better than PCSwap. This shows that combining multiple types of pairwise features is beneficial and our proposed methods are effective to leverage them.

To further analyze our results, we use the query selection scheme 4 and vary α to show the tradeoff between relevance improvement and query coverage of our algorithms. For PCSwap, we vary α and β to get the tradeoff. Figure 2 shows the tradeoff curves under different evaluation metrics. Note that we report relative gains against the base ranking function instead of absolute metric values. We do not directly show the values of α in the graph. Instead, for each α value, we show the gain (against the base ranking function) together with the number of affected queries. Intuitively, as the number of affected queries decreases (due to high confidence thresholds set by α), the relevance gain increases.

The results clearly show that PC and PFD significantly improve all the metrics compared to the base ranking function. The gains reported for PC and PFD are all statistically significant according to a Wilcoxon sign-rank test ($p \leq 0.001$). It is also clear that both PC and PFD outperform a simple re-ranker PCSwap over a wide range of query coverage.

Compared with PC, we find that PFD performs slightly better. One possible reason is because PC does not use the ranking scores of the base ranking function while PFD can exploit those scores and dynamically adjust the pairwise feature based prediction function.

These curves show that α is an effective knob to control the tradeoff. For example, if we want to make sure that we improve $NDCG_5$ by at least 10%, then we set $\alpha > 1.5$ and this will only affect +7% of queries (using PFD). If we want to have a large coverage of queries, we can lower the value so that our re-ranking methods can be triggered for more queries.

6.2.2 Query Selection Scheme Comparison

Figure 3 reports the performances among the four query selection schemes described in Section 5. For each scheme, we vary its parameter α with each giving us a tradeoff point between the relevance improvement and query coverage. A large α value means a lower query coverage but potentially higher relevance improvement. From this figure, we can see that all the 4 methods are demonstrated to be effective to identify queries with high potential gains. It is also clearly shown that the more sophisticated schemes (scheme 3 and 4) can outperform the simple ones (scheme 1 and 2) for both PC and PFD. Scheme 1 does not perform well compared to other schemes. This is because clicks are noisy: some clicks are due to perceived relevance (as opposed to landing page relevance). The results indicate that it is better to employ signals of multiple pairs of documents (scheme 3 and 4) than only one pair of documents (scheme 1 and 2).

6.2.3 Feature Effect Comparison

Figure 4 compares $NDCG_5$ for different combinations of pointwise features and true pairwise features used for training PC and PFD: all 100 features (denoted as PC and PFD), 30 true pairwise features with 10 concatenated pointwise features (denoted as PC_10_pnt and PFD_10_pnt), only 30 true pairwise features (denoted as PC_0_pnt and PFD_0_pnt), and only concatenated pointwise features (denoted as PC_only_pnt and PFD_only_pnt). From this figure, we can see that without the true pairwise features (PC_only_pnt and PFD_only_pnt), the performance significantly drops by only using the con-

catenated features like [15]. For PFD, it is clear that the addition of concatenated pointwise features boosts the performance. For PC, however, pointwise features do not affect the performance much. Overall, this shows that combining different types of pairwise relationships is important.

7. CONCLUSION

We have presented two novel machine learned re-ranking frameworks that can leverage multiple rich pairwise relationships between documents. Furthermore, we propose several schemes to estimate the potential gains of our re-ranking methods on each query and selectively apply them to queries with high confidence. We have demonstrated that our proposed re-ranking methods can significantly improve web search results for a commercial search engine and a recently proposed click log-based re-ranking algorithm. All these show the effectiveness of our methods.

In comparison to some previous work [6, 10, 21, 22], a distinctive feature of our re-ranking methods is their ability to model complex relationships among documents: different types of re-ranking signals are specified as features and the interaction between them is captured in our learning framework. This can inspire interesting research on pairwise features such as pairwise feature engineering and selection. In our re-ranking methods, we have fixed a base ranking function b when we learn a pairwise function h. An interesting direction we are investigating is a way of jointly training b and h.

8. REFERENCES

[1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of ACM SIGIR 2006*, pages 3–10, New York, NY, USA, 2006. ACM Press.

[2] N. Alon. Ranking tournaments. *SIAM J. Discret. Math.*, 20(1):137–142, 2006.

[3] C. Burges. Ranking as function approximation. *Algorithms for Approximation*, pages 3–18, 2006.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[5] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.

[6] M. L. Chao Liu and Y.-M. Wang. Post-rank reordering: Resolving preference misalignments between search engines and end users. In *CIKM '09: Proceedings of the 18th ACM international conference on Information and knowledge management*, 2009.

[7] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 1–10, New York, NY, USA, 2009. ACM.

[8] C. Cortes, M. Mohri, and A. Rastogi. Magnitude-preserving ranking algorithms. In *Proceedings of the 24th ICML*, 2007.

[9] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM'08: Proceedings of the international conference on Web search and web data mining*, pages 87–94, 2008.

[10] F. Diaz. Regularizing ad hoc retrieval scores. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 672–679, New York, NY, USA, 2005. ACM.

[11] G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 181–190, 2010.

[12] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[13] J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Statist.*, 29:1189–1232, 2001.

[14] J. Guiver and E. Snelson. Learning to rank with SoftRank and Gaussian processes. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008.

[15] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916, 2008.

[16] N. Jardine and C. J. V. Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 15:217–240, 1971.

[17] S. Ji, K. Zhou, C. Liao, Z. Zheng, G.-R. Xue, O. Chapelle, G. Sun, and H. Zha. Global ranking by exploiting user clicks. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 35–42, New York, NY, USA, 2009. ACM.

[18] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[19] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of ACM SIGIR 2005*, pages 154–161, New York, NY, USA, 2005. ACM Press.

[20] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2007.

[21] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global ranking using continuous conditional random fields. In *NIPS*, pages 1281–1288, 2008.

[22] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, W.-Y. Xiong, and H. Li. Learning to rank relational objects and its application to web search. In *WWW '08*, pages 407–416, 2008.

[23] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 77–86, New York, NY, USA, 2008. ACM.

[24] M. N. Volkovs and R. S. Zemel. Boltzrank: learning to maximize expected ranking gain. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1089–1096, New York, NY, USA, 2009. ACM.

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

[26] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th ACM SIGIR conference*, 2007.

[27] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, 2008.