Personalized Ranking Model Adaptation for Web Search

Hongning Wang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana IL, 61801 USA
wang296@illinois.edu

Xiaodong He[1], Ming-Wei Chang[1],
Yang Song[1], Ryen W. White[1], Wei Chu[2]
[1]Microsoft Research, Redmond WA, 98052 USA
[2]Microsoft Bing, Bellevue WA, 98004 USA
{xiaohe,minchang,yangsong,ryenw,wechu}
@microsoft.com

ABSTRACT

Search engines train and apply a single ranking model across all users, but searchers' information needs are diverse and cover a broad range of topics. Hence, a single user-independent ranking model is insufficient to satisfy different users' result preferences. Conventional personalization methods learn separate models of user interests and use those to re-rank the results from the generic model. Those methods require significant user history information to learn user preferences, have low coverage in the case of memory-based methods that learn direct associations between query-URL pairs, and have limited opportunity to markedly affect the ranking given that they only re-order top-ranked items.

In this paper, we propose a general ranking model adaptation framework for personalized search. Using a given user-independent ranking model trained offline and limited number of adaptation queries from individual users, the framework quickly learns to apply a series of linear transformations, e.g., scaling and shifting, over the parameters of the given global ranking model such that the adapted model can better fit each individual user's search preferences. Extensive experimentation based on a large set of search logs from a major commercial Web search engine confirms the effectiveness of the proposed method compared to several state-of-the-art ranking model adaptation methods.

Categories and Subject Descriptors

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

Keywords

Learning to rank, model adaptation, personalization

1. INTRODUCTION

Search engine users' information needs are diverse. Even for the same query, different users might express different preferences over the retrieved documents, resulting in distinct ranking requirements for search results [11, 27, 29]. For example, for the seemingly unambiguous query "facebook," some people might search for the login page of the social networking service, i.e., `www.facebook.com`; while others might be more interested about the recent news reports of the public company, stock quotes, and suchlike. However, the deployed ranking function in search engines is usually tuned according to general relevance judgments for a generic population of users [2, 19]; as a result, one globally-optimized ranking model cannot satisfy such diverse search preferences. In this scenario, it is preferable for the search engine to adapt the global ranking function to accommodate each individual user's result preference, i.e., personalized search.

Prior research has demonstrated that users' aggregated clicks are informative for learning their preferences and developing global search result ranking models [1, 17]. However, these models only reflect the common preferences across all searchers. Adapting the global ranking model towards each individual's search preferences to maximize search utility for each single user is more desirable. Existing personalization methods require rich user history information to learn user preferences [24, 25] (meaning that they are slow to adapt to user interests and interest dynamics), have low coverage in the case of memory-based methods that learn direct associations between query-URL pairs [28], and have limited opportunity to affect the ranking given that they frequently only re-order the top-ranked items [3]. In this paper, we proposed a method that effectively learns to adapt a generic ranking algorithm on a per-user basis, and overcomes many of the aforementioned challenges faced by existing personalization approaches.

Beside the adapted model's ranking accuracy, adaptation *efficiency* is also a primary consideration in this work. Existing work of ranking model adaptation in information retrieval (IR) mainly focuses on domain adaptation, e.g., from Web search to image search, where the goal of adaptation is to estimate a new ranking model for a target domain using information from a related source domain [6, 12, 13, 14]. Rooted in the classifier adaptation problem in transfer learning (c.f. [21]), the general assumption of domain adaptation in IR is that in the target domain there are insufficient labeled queries to accurately estimate a ranking model, but there is adequate supervision in the source domain. Therefore, to help model learning in the target domain, the adaptation methods need to effectively exploit supervision from the source domain. However, since most of existing methods estimate the adapted model in an offline manner, adaptation efficiency has received little attention in prior research. Nevertheless, in the scenario of adapting a generic ranking model for personalized search (our focus here), adaptation

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

efficiency becomes an equally important criterion for two main reasons: 1) such an operation must be executable on the scale of all the search engine users; 2) due to the dynamic nature of users' search intent and the need to offer searchers a great experience quickly, search engines cannot wait weeks or even days to collect adaptation data, since by then user preferences may have already shifted or they may have switched to another search engine. Our specific emphasis on adaptation efficiency prohibits us from directly applying most of existing domain adaptation methods.

Inspired by the linear regression based model adaptation methods widely studied in automatic speech recognition (e.g., maximum likelihood linear regression [18], minimum classification error linear regression [15]), we propose a general framework of ranking model adaptation, which enables rapid personalization. In particular, we assume that in a parametric ranking model different users' ranking preferences can be fully characterized by different settings of model parameters. For example, some users might prefer high authority websites, i.e., larger weight on the static rank score (page quality independent of query); while other users would emphasize query-term matching in documents, e.g., larger weight on retrieval-score features such as BM25. As a result, adjustment of the generic ranking model's parameters with respect to individual user's ranking preferences, e.g., click feedback, is necessary to satisfy their distinct ranking requirements. In our framework, such adjustment is achieved via linear transformations; and to meet the efficiency requirement for fast adaptation, we restrict such transformations to be simple and shared across features in a group-wise manner.

In this paper, based on the proposed framework for ranking model adaptation at the level of individual users, we aim to answer the following two research questions:

1) Under efficiency constraints, how should we effectively perform ranking model adaptation for each individual user? and;

2) What type of queries/users will benefit most from the proposed ranking model adaptation?

The first question is answered by the proposed ranking model adaptation framework, in which the parameters of a global ranking model are updated via a series of linear transformations, e.g., scaling and shifting, for each individual user. In the per-user basis ranking model adaptation scenario, the lack of adaptation data is a serious problem leading to sparse observations of ranking features for each user. To alleviate the sparsity problem, transformations are shared across features in a group-wise manner, such that it is possible to adapt the parameters of features that are not observed in the adaptation data. The proposed framework is general, and we demonstrated the detailed instantiation of the framework to three frequently used learning-to-rank algorithms, i.e., RankNet [4], LambdaRank [22] and RankSVM [16], where several important properties of the proposed adaptation framework is unveiled. To answer the second question, we collected a large set of search logs from Bing.com, and compared the proposed method against several state-of-the-art ranking model adaptation methods. Through extensive comparisons, our proposed method achieved significant improvement, not only in adaptation efficiency (measured in terms of the number of queries until reaches a performant state), but also in terms of the adapted model's ranking accuracy, against the baseline methods.

2. RELATED WORK

There are two major types of research closely related to our work in this paper, namely, ranking model adaptation and personalized search.

The major body of ranking model adaptation study in IR focuses on domain adaption, which can be categorized into three classes. One popular class is instance-based adaptation [6, 10, 13], which assumes certain parts of the data in the source domain can be reused for the target domain by re-weighting. Chen et al. [6] weighted the queries in the source domain by a heuristically defined utility function. In [13], Gao et al. employed a binary classifier to separate the documents in target domain from those in source domain, and then defined the importance of each source-domain document by the output of this classifier. The second category of work is feature-based [5], where a new feature representation is learned for the target domain and used to transfer knowledge across domains. Chen et al. proposed CLRank in [5], which constructs a new ranking feature representation so as to reduce the distributional difference between source and target domains. The third category is model-based [12, 14], which assumes the source and target ranking models share some parameters or priors. Geng et al. [14] regularized target-domain ranking model training using a given model from the source domain. Gao et al. [12] updated the source-domain model by the training errors on the adaptation data via stochastic gradient boosting algorithm.

To the best of our knowledge, few work attempts to adapt a generic ranking model for each individual user. Under efficiency constraints, both instance-based and feature-based methods are infeasible for this task, because they have to operate on numerous instances in the source domain, which is prohibitively expensive to perform for each single user. To avoid costly operation for each user, our proposed method falls in the class of model-based adaptation: we update the parameters of global ranking model for each individual user according to the observed click feedback. To alleviate the problem of sparse observation in adaptation data, transformations are shared across features so that parameters of unseen features can also be effectively updated.

The task of personalized search aims at leveraging information about an individual to identify the most relevant search results for them. Mainstream of personalization techniques target the extraction of user-centric profiles or features, e.g., location, gender and click history, and incorporating such information into the original ranking function. Teevan et al. encoded user profiles extracted from relevance feedback to re-rank the retrieved documents [26]. Dou et al. [9] performed a large-scale evaluation of several personalized search strategies, e.g., user profile based re-ranking [7], and revealed that personalization has mixed effects on the ranking performance. These and other personalization models (e.g., [24, 25]) use significant quantities of search history to learn interest profiles for each user, requiring sufficient data available to perform personalization effectively.

Memory-based personalization techniques learn direct associations between query-URL pairs [28] (e.g., given this query, the current user consistently selects a particular URL), which can perform well given high revisitation likelihoods, but have limited query coverage. Shen et al. [23] developed a context-sensitive language model by introducing both click feedback and preceding queries for short-term personalization. However, these short-term models are specific to the current context and cannot generalize well to accommodate

user's general preferences. Once a model is learned, a common strategy for the *application* of personalization is to re-rank the top-n results [3, 9]. This means the personalized models do not have the opportunity to promote results of low general interest (i.e., outside of the top n), but of high interest to the current user, into the top-ranked results.

In our approach, we quickly adapt the search engine's generic ranking function on a per-user basis, and therefore overcome many of these shortcomings.

3. RANKING MODEL ADAPTATION

Inspired by the linear regression based model adaptation methods in speech recognition [15, 18], we propose a general framework to perform ranking model adaptation. We assume that a global ranking model is trained based on a large user-independent training set. For each user, an adapted model is obtained by applying a set of learned linear transformations, e.g., scaling and shifting, to the parameters of the global model based on each individual user's adaptation data, e.g., query with corresponding clicks.

In the following discussions, we first describe our general framework of ranking model adaptation, and then we take three frequently used learning to rank algorithms, i.e., RankNet [4], LambdaRank [22] and RankSVM [16], as examples to demonstrate the detailed procedures of applying the proposed adaptation framework.

3.1 General Framework

For a given set of queries $Q^u = \{q_1^u, q_2^u, \ldots, q_m^u\}$ from user u, each query q_i^u is associated with a list of document-label pairs $\{(x_{i1}^u, y_{i1}^u), (x_{i2}^u, y_{i2}^u), \ldots, (x_{in}^u, y_{in}^u)\}$, where x_{ij}^u denotes a retrieved document represented by a V-dimensional vector of ranking features, and y_{ij}^u is the corresponding relevance label indicating if the document x_{ij}^u is relevant to user u (e.g., clicks). Since our focus of this work is on user-level ranking model adaptation, in the following discussions we ignore the superscript u to make the notations concise when no ambiguity is involved.

A ranking model f is defined as a mapping from a document x_{ij} to its ranking score s_{ij}, i.e., $f : x_{ij} \rightarrow s_{ij}$, such that when we order the retrieved documents for query q by f, a certain ranking metric, e.g., mean average precision (MAP) or precision at k (P@k) [2], is optimized. Such ranking model can be manually set, or estimated by an automatic algorithm based on a collection of annotated queries [19]. In this work, we focus on *linear* ranking models, which can be characterized by a parametric form of linear combination of ranking features, i.e., $f(x) = w^{\mathsf{T}} x$, where w is the linear coefficients for the corresponding ranking features.

Denoting $f^s(x) = {w^s}^{\mathsf{T}} x$ as the given global ranking model estimated in a user-independent manner, the adaptation of $f^s(x)$ for each individual user is performed via the linear transformations defined by a $V \times (V+1)$ dimensional matrix A^u, by which three linear operations, i.e., scaling, shifting and rotation, can be encoded. More precisely,

$$f^u(x) = (A^u \tilde{w}^s)^{\mathsf{T}} x \tag{1}$$

where $\tilde{w}^s$ is an augmented vector of w^s, i.e., $\tilde{w}^s = (w^s, 1)$, to facilitate the shifting operation for parameter adaptation.

There are two major considerations in designing such a transformation matrix A^u. First, a full transformation matrix has the number of $O(V^2)$ free parameters, which is redundant and even larger than the number of parameters needed to estimate a new ranking model for each user (i.e., $O(V)$). As a result, it is infeasible for us to estimate a full transformation matrix for every user. To reduce the size of free parameters in A^u, we decide to only focus on the scaling and shifting operations for adapting the parameters in $f^s(x)$. This reduces the size of free parameters in A^u from $O(V^2)$ to $O(V)$. Second, a more important consideration is how to alleviate the problem of sparse observation of ranking features in the limited adaptation data. Because some advanced ranking features used in modern search engines, e.g., topic category of documents, might not be trigged in the scattered adaptation queries, one will encounter missing feature values. In order to properly update the parameters for unseen features during adaptation, we organize the features in groups and share the same shifting and scaling transformations to the parameters within the same group.

Based on the above considerations, we design the transformation matrix A^u to be the following specific form,

$$\mathbf{A}^u = \begin{pmatrix} a^u_{g(1)} & 0 & \cdots & b^u_{g(1)} \\ 0 & a^u_{g(2)} & \cdots & b^u_{g(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a^u_{g(V)} & b^u_{g(V)} \end{pmatrix}_{V \times (V+1)}$$

where $g(\cdot)$ is a feature grouping function, which maps V original ranking features to K different groups, a_k^u and b_k^u denote the scaling and shifting operations applied to the linear coefficients w^s of the source model $f^s(x)$ in group k. As a result, Eq (1) can be realized as,

$$f^u(x) = \sum_{k=1}^{K} \sum_{g(i)=k} (a_k^u w_i^s + b_k^u) x_i \tag{2}$$

The grouping function $g(\cdot)$ defines the transformation sharing among the original ranking features. It enables the observations from seen features to be propagated to unseen features within the same group during adaptation, which is critical in addressing the problem of sparsity in the limited adaptation data. However, defining the optimal grouping of ranking features is non-trivial; we postpone the discussion of constructing $g(\cdot)$ to Section 3.4.

Once the grouping function $g(\cdot)$ is given, another important component in our adaptation framework is the criterion to estimate the optimal transformation matrix A^u. An ideal transformation should be able to adjust the generic ranking model to meet each individual's ranking preference, i.e., maximizing the search utility for each user. In the study of learning-to-rank algorithms in IR, various types of objective functions, e.g., pairwise and listwise, have been proposed to realize the goal of optimizing ranking metrics [19]. Therefore, to make the proposed framework generally applicable, we do not restrict our adaptation objective to any specific form, but instantiate it with the objective function from the ranking algorithm we choose to adapt.

We want to emphasize that although in our framework we utilize the objective function from the ranking algorithm to be adapted as the criterion to estimate the transformation matrix A^u, it does not necessarily restrict the global model to being estimated by the same ranking algorithm. As long as the global model and adapted model share the same model structure, e.g., neural network structure in RankNet and linear model in RankSVM, the proposed adaptation framework is applicable.

To summarize, our general framework for ranking model

adaptation can be formalized as follows,

$$\min_{A^u} \quad L_{\text{adapt}}(A^u) = L(Q^u; f^u) + \lambda R(A^u) \tag{3}$$
$$\text{where} \quad f^u(x) = (A^u \tilde{w}^s)^\intercal x \text{ and } \tilde{w}^s = (w^s, 1)$$

in which $L(Q^u; f^u)$ is the objective function defined in the ranking algorithm we choose to adapt, e.g., cross-entropy in RankNet or hinge loss in RankSVM, $R(A^u)$ is a regularization function defined on the transformation matrix A^u, λ is a trade-off parameter, and w^s is the linear coefficients for ranking features in the global ranking model.

3.2 Adapting RankNet & LambdaRank

RankNet [4] is a probabilistic learning-to-rank algorithm, which models the probability that a document x_{ij} is ranked higher than x_{il} for query q_i, i.e., $P(y_{ij} > y_{il})$. A logistic function is employed to map the predicted ranking scores of two documents, e.g., s_{ij} and s_{il}, to probability of ordering,

$$P(y_{ij} > y_{il}) = \frac{1}{1 + e^{-(s_{ij} - s_{il})}}.$$

The training objective function in RankNet is defined as the cross-entropy between the predicted pairwise ordering probabilities and the observed pairwise preferences in the training data, i.e.,

$$\begin{aligned} L_{\text{RankNet}} = &\sum_{q_i} \sum_{y_{ij}, y_{il}} -\bar{P}(y_{ij} > y_{il}) \log P(y_{ij} > y_{il}) \\ &- (1 - \bar{P}(y_{ij} > y_{il})) \log(1 - P(y_{ij} > y_{il})) \end{aligned} \tag{4}$$

where $\bar{P}(y_{ij} > y_{il})$ is the empirically estimated probability that x_{ij} is ranked higher than x_{il}.

RankNet is usually optimized via a neural network. Because in each layer of a neural network, every neuron's output is linearly combined to feed into the next layer, our adaption framework can be smoothly applied to the linear weights for each neuron (e.g., different transformation matrices for each neuron in the hidden layers). In order to understand the effect of the proposed adaptation in RankNet, we will use a RankNet with no hidden layers for discussion, but the same procedure can be applied to general RankNet with an arbitrary number of hidden layers.

To adapt RankNet, we take the same cross-entropy function defined in Eq (4) as our adaptation objective, and define the following regularization function on matrix A^u,

$$R(A^u) = \frac{1}{2} \sum_{k=1}^{K} (a_k^u - 1)^2 + \frac{\sigma}{2} \sum_{k=1}^{K} b_k^{u2} \tag{5}$$

where we penalize the transformation which increases the discrepancy between the adapted model and the global model, and σ is a parameter that controls the balance between the penalty on shifting and scaling operations.

As a result, the gradient with respect to the scaling parameter a_k^u can be calculated as,

$$\begin{aligned} &\frac{\partial L_{\text{adaptRankNet}}(A^u)}{\partial a_k^u} \\ &= \sum_{q_i \in Q_u} \sum_{y_{ij} > y_{il}} [P(y_{ij} > y_{il}) - 1] \frac{\partial (A^u \tilde{w}^s)^\intercal \Delta x_{ijl}}{\partial a_k^u} + \lambda \frac{\partial R(A^u)}{\partial a_k^u} \\ &= \sum_{q_i \in Q_u} \sum_{y_{ij} > y_{il}} [P(y_{ij} > y_{il}) - 1] \sum_{g(v)=k} w_v^s \Delta x_{ijlv} + \lambda (a_k^u - 1) \end{aligned} \tag{6}$$

where Δx_{ijl} is a V-dimensional vector defined as $\Delta x_{ijl} = x_{ij} - x_{il}$. Accordingly, the gradient with respect to b_k^u is,

$$\begin{aligned} &\frac{\partial L_{\text{adaptRankNet}}(A^u)}{\partial b_k^u} \\ &= \sum_{q_i \in Q_u} \sum_{y_{ij} > y_{il}} [P(y_{ij} > y_{il}) - 1] \sum_{g(v)=k} \Delta x_{ijlv} + \lambda \sigma b_k^u \end{aligned} \tag{7}$$

The above gradients induce a new neural network defined over the linear transformations, where the connection among neurons is specified by the grouping function $g(\cdot)$: the term $P(y_{ij} > y_{il}) - 1$ in Eq (6) and Eq (7) represents the prediction error of the global ranking model on the adaptation data; and based on this error, the gradients specify the direction in which the adaptation should take. We can note that the gradients for a_k^u and b_k^u are estimated based on all the observations of ranking features in the same group; as a result, the parameters for the unseen features can also get updated, by sharing such jointly estimated transformations.

To generalize this procedure to RankNet with multiple hidden layers, we only need to replace the error term defined by $P(y_{ij} > y_{il}) - 1$ with the corresponding back-propagation error in each hidden layer in Eq (6) and Eq (7), and the original optimization procedure for RankNet can be directly applied to the adapted problem. One thing we should note is that since one can set different number of neurons in each hidden layer, to apply the proposed adaptation in a RankNet with multiple layers, we need to specify the grouping function $g(\cdot)$ for each neuron in the hidden layers. This can be achieved via the clustering method proposed in Section 3.4.

Based on the discussion of adapting RankNet within the proposed framework, it is straightforward to adapt LambdaRank [22] in a similar manner. As a listwise learning-to-rank algorithm, LambdaRank modifies the error term in RankNet by adding an additional correction term and names such modified error as lambda function,

$$\lambda_{ijl} = [P(y_{ij} > y_{il}) - 1] |\Delta_{\text{IR-Metric}}| \tag{8}$$

where $|\Delta_{\text{IR-Metric}}|$ is the change of any specific ranking metric, e.g., MAP or NDCG, given by swapping the rank positions of document x_{ij} and x_{il} while leaving the rank positions of all other documents unchanged.

Therefore, to adapt LambdaRank within our framework, we only need to replace the error function of the output layer in RankNet with the lambda function defined in Eq (8), and all the other procedures remain the same as in RankNet.

3.3 Adapting RankSVM

RankSVM [16] is a classic pairwise learning-to-rank algorithm, in which the learning problem is formalized as,

$$\begin{aligned} \min_{w, \xi_{ijl}} \quad & \frac{1}{2} ||w||^2 + C \sum_{q_i} \sum_{j,l} \xi_{ijl} \\ \text{s.t.} \quad & w^\intercal \Delta x_{ijl} \geq 1 - \xi_{ijl}, \forall q_i, x_{ij}, x_{il} \\ & \xi_{ijl} \geq 0 \\ \text{where} \quad & y_{ij} > y_{ij} \text{ and } \Delta x_{ijl} = x_{ij} - x_{il} \end{aligned} \tag{9}$$

where C is a trade-off parameter to control the balance between model complexity and empirical hinge loss over the identified preference pairs from the training data.

To adapt RankSVM, we keep the hinge loss defined in Eq (9) as our adaptation objective, and use the same regularization function for A^u defined in Eq (5). By taking the linear transformation $w^u = A^u \tilde{w}^s$ into Eq (9), we get the

adapted problem for RankSVM as,

$$\min_{\mathbf{a}^u,\mathbf{b}^u,\xi_{ij}} \quad \frac{1}{2}\sum_{k=1}^{K}(a_k^u-1)^2+\frac{\sigma}{2}\sum_{k=1}^{K}(b_k^u)^2+C\sum_{q_i}\sum_{j,l}\xi_{ijl} \quad (10)$$

$$\text{s.t. } {w^u}^{\mathsf{T}}\Delta x_{ijl}\geq 1-\xi_{ijl},\forall q_i,x_{ij},x_{il}$$

$$\xi_{ijl}\geq 0$$

$$\text{where } w^u=A^u\tilde{w}^s,\ y_{ij}>y_{il},\text{ and } \Delta x_{ijl}=x_{ij}-x_{il}$$

Since the input for RankSVM training is document pairs, in the following discussion, we briefly denote Δx_{ijl} as $\vec{x}_t$, in which the subscript t ranges over all the preference pairs in the adaptation set, to simplify the notations. Following the conventional derivation of RankSVM, we get the dual problem of Eq (9) by introducing a set of Lagrange multipliers α,

$$\max_{\alpha}\sum_{t}\Big[1-f^s(\vec{x}_t)\Big]\alpha_t-\frac{1}{2}\alpha^{\mathsf{T}}\Big[K_1(\vec{\mathbf{x}},\vec{\mathbf{x}})+K_2(\vec{\mathbf{x}},\vec{\mathbf{x}})\Big]\alpha \quad (11)$$

$$s.t.\ 0\leq\alpha_t\leq C,\forall t$$

$$\text{where } K_1(\vec{x}_t,\vec{x}_r)=\sum_{k=1}^{K}\Big(\sum_{g(v)=k}w_v^s\vec{x}_{tv}\Big)\Big(\sum_{g(v)=k}w_v^s\vec{x}_{rv}\Big)$$

$$K_2(\vec{x}_t,\vec{x}_r)=\frac{1}{\sigma}\sum_{k=1}^{K}\Big(\sum_{g(v)=k}\vec{x}_{tv}\Big)\Big(\sum_{g(v)=k}\vec{x}_{rv}\Big)$$

By solving the above dual problem, we can get the optimal transformations as,

$$a_k^u=1+\sum_t\alpha_t\sum_{g(v)=k}w_v^s\vec{x}_{tv}$$

$$b_k^u=\frac{1}{\sigma}\sum_t\alpha_t\sum_{g(v)=k}\vec{x}_{tv}$$

The effect of the proposed adaptation on RankSVM is clearly depicted in its dual form. First, as we know that the linear coefficients in front of the Lagrange multipliers α in Eq (11) correspond to the separation margin for each training instance in SVM. In the adapted problem, the margin is rescaled according to the global model $f^s(\vec{x}_t)$'s prediction on the adaptation data: if the global model can well separate the adaptation pair $\vec{x}_t$, i.e., $f^s(\vec{x}_t)>0$, the margin decreases, indicating this case is not crucial for adaptation; if the global model fails to correctly predict the order for this pair, i.e., $f^s(\vec{x}_t)\leq 0$, the margin increases, and $\vec{x}_t$ becomes a more important instance in adaptation for this particular user. This precisely interprets the effect of model-based adaptation: we only update the global model when it makes a mistake on the adaptation data; otherwise keep it intact. Second, the proposed linear transformations induce two new kernels in a compressed space: $K_1(\vec{x}_t,\vec{x}_s)$, corresponding to the scaling operation, defines a compound polynomial kernel over the ranking features projected by the global ranking model w^s; and $K_2(\vec{x}_t,\vec{x}_s)$, corresponding to the shifting operation, defines another compound polynomial kernel over the original ranking features. Both kernels work in a compressed K-dimensional space determined by the feature group mapping function $g(\cdot)$, and are interpolated by the balance parameter σ between the regularizations for shifting and scaling operations. As a result, non-linearity is introduced to the original linear RankSVM model, and such non-linearity helps the model to leverage the observations from seen features to the unseen ones in the same group.

3.4 Feature Grouping

In the proposed framework, a feature grouping function $g(\cdot)$ is used to organize the ranking features so that shared transformation is performed on the parameters of features in the same group. This grouping can be given *a priori* according to the design of ranking features, or be determined by data-driven approaches based on a given set of queries and documents. In this work, we proposed and compared three possible ways of creating such feature groups.

The first grouping method is based on the name of ranking features. Ranking features are usually described by the way they are generated, e.g., *BM25 of Body*, *BM25 of Title* [20], such that the name of a ranking feature provides informative indication of its functionality. Given the naming scheme of features in a collection, we can manually define patterns to cluster the features into groups. We denote this grouping method *Name*.

The second method is based on the co-clustering algorithms in document analysis. Similar to [8], we first project the document-feature matrix into a lower dimensional space by singular value decomposition (SVD), and then perform k-means clustering to group the features into K clusters based on this low dimensional representation. We name this grouping method *SVD*.

The third method groups features by the corresponding learned parameters in the ranking models. We first evenly split the training collection into N non-overlapping folds, and train a single ranking model, e.g., RankSVM, on each fold. Then, we create a $V\times N$ matrix by putting the learned parameters from those N independent models together, on which k-means clustering is applied to extract K feature groups. We name this grouping method *Cross*.

The *Name* method requires the collection to have a reasonable feature naming scheme; if the ranking features are arbitrarily named, e.g., named by ID, such method cannot be used. The *SVD* method is generally applicable since it only requires a collection of query-document pairs represented by the ranking features. In the *Cross* method, besides a set of documents, a relevance judgment for each document with respect to a given query is also needed to estimate the grouping of features. In particular, for RankNet with multiple layers, the *Cross* method can be used to estimate the grouping function for each neuron in the network based on the learned weights of connections.

3.5 Discussion

There are four advantages of the proposed adaptation framework. First, it is a general framework for ranking model adaptation, which is applicable to a majority of existing learning-to-rank algorithms [19]. Second, since the proposed adaptation framework is model-based, unlike the instance-based or feature-based adaptation methods, it does not need to operate on the numerous data from the source domain, which makes the per-user basis ranking model adaptation feasible. Third, the same optimization technique for the original learning algorithm can be directly applied with little change, so that it does not increase the complexity of solving the adaptation problem. And in the adaptation phase, we only need to solve the optimization problem over a small amount of adaptation data, which ensures the computational efficiency for performing the adaptation on the scale of all search engine users. Fourth, and most importantly, transformation is shared across features in the proposed adaptation framework. According to Eq (2), the same

transformation is applied onto the parameters of features in the same group, which renders several important properties in the adapted ranking models: in RankNet, the gradients for scaling (in Eq (6)) and shifting (in Eq (7)) operations are estimated based on all the observations in the same group; while in RankSVM, two new non-linear kernels are induced over the original linear function space. As a result, even though we might not observe a specific feature occurring in the adaptation data, we can still propagate the information from other features in the same group to update it properly.

4. EXPERIMENT

In order to evaluate the proposed adaptation framework, we performed a series of experiments on large-scale search query logs sampled from Bing.com. A set of state-of-the-art ranking model adaptation methods were included as baselines to validate the effectiveness of the proposed method.

4.1 Dataset and Settings

We extracted five days' search logs from May 27, 2012 to May 31, 2012 from Bing.com for our experiments. During this period, a subset of users were randomly selected and all their search activities were collected, including the anonymized user ID, query string, timestamp, top 10 returned document lists and the corresponding clicks. The queries were ordered by their timestamp in each user, and the documents were sorted by their original order returned by the search engine under each query.

To apply the proposed adaptation method and compare with the baselines according to user click feedback, we can only use the queries with clicks. Therefore, in our experiment, we filtered out the queries without clicks and required each user to have at least two queries with clicks, i.e., one for adaptation and one for testing.

We also sampled a large set of manually annotated query logs from our existing data collection as the user-independent training set for adaptation. Each query-document pair in this annotation set is labeled with a five-grade relevance score, i.e., from 0 for "bad" to 4 for "perfect." Documents in both the selected user data set and annotation data set are represented by a set of 1,830 ranking features selected from their overlapped feature set, including frequently used ranking features such as BM25, language model score and PageRank. Using the language of domain adaptation, we treat the collection of annotated queries as our source domain and each user's queries with clicks as target domain. This setting provides a good simulation for real Web search scenario, where the generic rankers in use are usually trained on offline annotated data, and thus it helps us compare the effectiveness of different ranking model adaptation methods. The basic statistics of the annotation set and selected user set are summarized in Table 1.

Preference pairs are extracted from user's clicks to reflect their unique search requirements. In order to ameliorate the positional biases inherent in click data [1], we followed Joachims et al.'s method to extract the click preference pairs [17]. In particular, we employed two click heuristics: for a given query q with a ranked document list $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ returned by the search engine,

1. "*Click* $\succ$ *Skip Above*": extract preference pair $x_i \succ x_j$ for all pairs $1 \leq j < i$ with $y_i > y_j$.
2. "*Click* $\succ$ *Skip Next*": extract preference pair $x_i \succ x_{i+1}$ for all $y_i > y_{i+1}$.

Table 1: Statistics of annotation and user data set.

	# Users	# Queries	# Documents
Annotation Set	-	49,782	2,320,711
User Set	34,827	187,484	1,744,969

In order to avoid defining different feature grouping functions for different ranking algorithms we selected to adapt, e.g., in RankNet each neuron in the hidden layers needs a possibly different grouping function but in RankSVM only one grouping function is needed for the original features, we decided *not* to use hidden layers in the neuron networks for RankNet and LambdaRank in our experiment. As a result, the same grouping function defined on the original ranking features can be directly used in RankNet, LambdaRank and RankSVM. A LambdaRank model optimizing NDCG@10 is trained on the annotation set and used as the global ranking model for adaptation in the following experiments (denoted as *Source-Only*)[1]. The trade-off parameter λ (in Eq (3)) and σ (in Eq (5)) in our method are selected by 5-fold cross validation on the whole user set in advance.

To quantitatively compare different adaptation methods' performance, we employed a set of standard IR evaluation metrics: by treating all the clicked documents as relevant, we calculated Mean Average Precision (MAP), Precision at 1 (P@1), Precision at 3 (P@3) and Mean Reciprocal Rank (MRR). Definitions of these metrics can be found in [2].

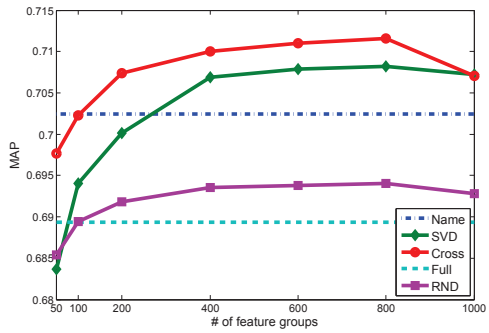
4.2 Analysis of Feature Grouping

The grouping of features has a substantial impact on the adaptation performance in our method, since transformations will be shared for the parameters of features in the same group. Ideally, we should put parameters that need to be updated synchronously in the same group. In this experiment, we evaluated the three feature grouping methods, i.e., *Name*, *SVD*, and *Cross*, proposed in Section 3.4. For comparison purposes, we also included two trivial grouping methods: 1) "*Full*," which creates a group for every single feature, i.e., no transformation is shared across features; 2) "*RND*," which randomly allocates features into K groups.

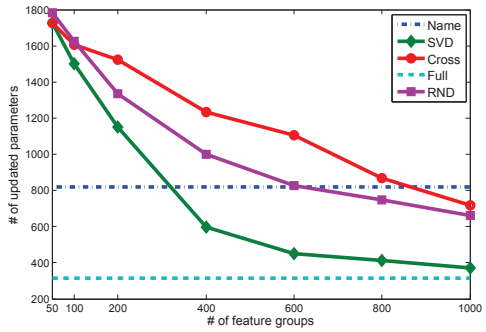
In our data set, according to the naming scheme of features, i.e., *featureType_source_seqID*, 413 feature groups are extracted by the *Name* method. The two data-driven approaches, *SVD* and *Cross*, were performed on the annotation set, but we have to specify the group size K for them in advance. To analyze the effect of group size K in our proposed adaptation framework, we evaluated the adaptation performance of RankNet by varying the setting of K. To control the number of adaptation queries in each user, which influences the adaptation performance, we selected a subset of users, where each user has at least six queries with clicks (close to the average number of queries with clicks per user in our collection), and used the first three queries for adaptation and last three queries for testing in each user. This leads to a collection of 8,879 users with 112,069 queries.

The MAP ranking performance of adapted RankNet with different feature grouping methods is shown in Figure 1 (a). First, it is clear that a properly set K is crucial for both *SVD* and *Cross* methods. The more groups we set, the more adaptation parameters need to estimate based on the limited adaptation data; but if we set too few feature groups, the

[1]Since we only used a subset of annotated queries and features, the results here do not reflect the actual performance of the search engine.



(a) Adaptation performance with different K



(b) Number of updated parameters with different K

Figure 1: Analysis of feature grouping in RankNet adaptation.

discriminations among the features will be lost due to inaccurate parameter updating by adaptation sharing. Besides, Figure 1 (a) also shows that the adaptation performance is less sensitive to K around its optimal value, i.e., the performance as indicated by MAP is stable in a wide range of K from 400 to 800, for both *SVD* and *Cross*.

Another observation in Figure 1 (a) is that *Cross* performed consistently better than the other grouping methods under the same setting of K. Because in the *Cross* method features with similar contributions (i.e., linear coefficients) to document ranking are grouped together, and they tend to update synchronously. Sharing transformations among such features is more desirable. In contrast, other grouping methods cannot exploit such relationship among the features, e.g., *SVD* only exploits the co-occurrence relationship between features, and thus they achieved worse results.

In order to understand the in-depth effect of feature grouping in our adaptation framework, we computed the average number of updated parameters in the adapted ranking model for each user with respect to different group size K and illustrated the results in Figure 1 (b). We can note that on average only 316 features (with a standard deviation of 214) can be observed in the adaptation data according to the result of *Full* method. However, because of adaptation transformation sharing across features in our framework, the number of parameters that have been actually adjusted is much larger. For example, with 800 groups, about 870 parameters (with a standard deviation of 220) on average are effectively updated by the *Cross* method, indicating that more than 60% of updated parameters are adapted without actual observations. On the other hand, when K becomes smaller, the number of updated parameters increases rapidly. Consequently, using too fewer groups forces less relevant features get updated by the shared transformations, which in turn degrades the overall adaptation performance.

Similar adaptation results with respect to group size K were also observed in LambdaRank and RankSVM. In the following experiments, to avoid selecting K for each individual user and the variation of performance introduced by this factor, we fix K to be 800 for both *SVD* and *Cross*.

4.3 Comparison of Adaptation Performance

4.3.1 Baselines

To make a thorough evaluation of the proposed adaptation method, we included several state-of-the-art ranking model adaptation methods as baselines, covering instance-based, feature-based and model-based methods, for comparisons. We describe the employed baselines briefly in text below.

TransRank [6] is an instance-based ranking model adaptation method, in which a utility function is defined to select the top k important queries from the source domain into target domain for model training. IW-RankSVM [13] is another instance-based adaptation method, which re-weights the instances in source domain by measuring its distance to the classification hyperplane between source and target domain, and only uses those re-weighted instances from source domain for target-domain model training. CLRank [5] is a feature-based adaptation method, which constructs a new joint feature representation for both source and target domains to reduce the distributional difference between them.

However, it would be prohibitively expensive if we directly applied these baselines for every user, because such methods need to access all the offline training data during adaptation. To make these methods applicable in our application scenario, we pooled all the user's adaptation data together to form a combined user collection, on which the above baseline methods are applied. In addition, we also trained a new LambdaRank model optimizing MAP on this integrated user collection as a baseline, and named it Target-Only.

RA-RankSVM [14] is a model-based adaptation method, which treats the source-domain ranking model as an additional regularization for model training in the target domain. Based on RA-RankSVM, we used the same regularization idea in RankNet and LambdaRank to get the corresponding RA-RankNet and RA-LambdaRank baselines. Besides, without knowledge about the global model, we estimated a ranking model only based on each individual user's adaptation data, and denoted such method as Tar, e.g., Tar-RankSVM, accordingly.

All baseline methods' hyper-parameters, e.g., trade-off parameter C in RA-RankSVM, are tuned by 5-fold cross validation on the full user data set in advance.

4.3.2 Adaptation Accuracy

We performed the experiment on all user data in our collection, in which the first 50% of queries from each user are used for adaptation and the rest are used for testing.

• **Comparison in per-user basis adaptation:** we first compared the ranking performance of our proposed adaptation methods (under all the three grouping methods) with the model-based adaptation baseline methods, e.g., RA-RankSVM, RA-RankNet and RA-LambdaRank, and the baseline methods solely depend on the adaptation data, i.e., Tar-RankSVM, Tar-RankNet and Tar-LambdaRank. These are the only baselines applicable in the scenario of per-user basis ranking model adaptation. In particular, MAP metric is chosen to be optimized in the adapted LambdaRank model.

Table 2: Comparison of per-user basis ranking model adaptation performance.

	MAP	P@1	P@3	MRR
Tar-RankSVM	0.6240	0.4905	0.2335	0.6282
RA-RankSVM	0.6366	0.4809	0.2510	0.6410
Name-RankSVM	0.6544	0.5078	0.2546	0.6585
SVD-RankSVM	**0.6643**	**0.5209**	**0.2579**	**0.6687**
Cross-RankSVM	0.6638	0.5200	0.2574	0.6681
Tar-RankNet	0.6342	0.5051	0.2360	0.6384
RA-RankNet	0.6577	0.5267	0.2481	0.6619
Name-RankNet	0.6709	0.5425	0.2535	0.6750
SVD-RankNet	0.6751	0.5412	0.2581	0.6796
Cross-RankNet	**0.6781**	**0.5450**	**0.2593**	**0.6826**
Tar-LambdaRank	0.6436	0.5182	0.2384	0.6477
RA-LambdaRank	0.6616	0.5341	0.2479	0.6657
Name-LambdaRank	0.6814	0.5556	0.2569	0.6859
SVD-LambdaRank	0.6878	0.5590	0.2616	0.6925
Cross-LambdaRank	**0.6922**	**0.5662**	**0.2629**	**0.6969**

In Table 2, we can observe significant improvements in ranking performance from the model-based adaptation methods, i.e., our methods and RA methods, against the methods solely depending on the adaptation data, i.e., Tar methods. As discussed before, sparsity is a serious problem in the per-user basis model estimation. Tar methods cannot estimate the parameters for the unseen features, and thus their ranking capability is limited. RA methods alleviate such deficiency by using the global model as back-off: for features not observed in the adaptation data, the parameters from the global model would be used. In our proposed method, besides back-off to the global model when no observation is available (as shown in Eq (5)), we also propagate the observations from seen features to unseen features by transformation sharing to help the model better estimate the parameters of those unseen features. As a result, our adaptation methods, under all grouping methods, outperformed the corresponding RA adaptation methods (all the improvements are significant with p-value<0.01 under paired t-test).

Another observation in Table 2 is that the adapted LambdaRank performed consistently better than the adapted RankNet and RankSVM within our framework. In LambdaRank the lambda function helps the model directly optimize the IR-related metrics, e.g., MAP in our case, while RankNet and RankSVM can only minimize pairwise loss. LambdaRank has shown better performance than those pairwise learning-to-rank algorithms in many classical ranking tasks [22]. In our adaptation framework, such advantage of LambdaRank is preserved since the same lambda function and optimization procedure are applied as in the original LambdaRank. This demonstrates the flexibility of our adaptation framework, in which we can choose to adapt any specific ranking algorithm according to the property of the task.

• **Comparison with integrated adaptation:** according to the results in Table 2, we compared our best performing method *Cross*-LambdaRank with the instance-based and feature-based ranking model adaptation methods, and list the results in Table 3.

First of all, we can notice that in Table 3 the global ranking model trained on the annotation set (i.e., Source-Only) did not perform well on the user testing set; while the model trained on the integrated user data improved most of the ranking metrics over 10%. This indicates evident distributional difference between the generic annotation set and user click set. Through instance re-weighting, i.e., IW-RankSVM and TransRank, or feature construction, i.e., CLRank, all baseline adaptation methods achieved improved results against the global model. For these baseline methods, since we have pooled all the users' adaptation data together, sparsity is no longer a serious problem. However, individual user's specific ranking preferences will be overwhelmed once we pooled different users' clicks together. In our adaptation method, e.g., *Cross*-LambdaRank, the global model is adapted for each individual user towards maximizing the search utility based on their own adaptation data. As a result, *Cross*-LambdaRank outperformed all these baseline adaptation methods, which are originally designed for domain adaptation, in this user-oriented evaluation.

Table 3: Comparison of adaptation performance.

	MAP	P@1	P@3	MRR
Source-Only	0.5637	0.3356	0.2503	0.5679
Target-Only	0.6258	0.4667	0.2468	0.6298
IW-RankSVM	0.6427	0.4865	0.2506	0.6470
TransRank	0.6468	0.5202	0.2400	0.6512
CLRank	0.6590	0.5090	0.2561	0.6594
Cross-LambdaRank	**0.6922**	**0.5662**	**0.2629**	**0.6969**

• **Query-/User-level improvement analysis:** the results shown in Table 2 and Table 3 are averaged over all the users' testing queries. It is necessary to further investigate to what extent and what types of users/queries can benefit from the proposed adaptation method. We analyzed the detailed ranking results given by *Cross*-LambdaRank against those from the global ranking model and RA-LambdaRank, which is the best baseline according to Table 2 and Table 3.

Table 4: Ranking performance gain against the global model from *Cross*-LambdaRank and RA-LambdaRank on repeated and non-repeated queries.

Query Type	Method	ΔMAP	ΔP@1	ΔP@3	ΔMRR
Repeated	RA	0.2329	0.4082	0.0249	0.2331
	Cross	0.2375	0.4129	0.0273	0.2379
Non-repeated	RA	-0.0337	-0.0050	-0.0292	-0.0342
	Cross	0.0204	0.0498	-0.0024	0.0206

Query repetition is a common phenomenon in user's query log, and it is crucial for many memory-based personalization methods [9, 28]. First, we categorized the testing queries as repeated queries, if it occurred in the corresponding user's adaptation query set, and the rest as non-repeated ones; and then computed the improvement of ranking performance against the global model from *Cross*-LambdaRank and RA-LambdaRank on these two types of testing queries. As shown in Table 4 (all the differences are significant with p-value<0.01 under paired t-test), both methods achieved notably improvements against the global model on the repeated queries, but only *Cross*-LambdaRank attained improved results on the non-repeated queries. For the repeated queries, both methods can simply "memorize" and "promote" the documents clicked by the user; while for the non-repeated queries, because RA-LambdaRank did not get any direct observations to adjust the relevant ranking features, it could not generalize well from the adaptation data. In *Cross*-LambdaRank, the unseen features can also be updated via transformation sharing, which affords the model a better ranking capability on those non-repeated queries.

In addition, we also applied a proprietary multi-label classifier to annotate the query intent into 63 categories, e.g., navigational, commerce and etc., and found that the major improvement of our method against the global model comes from the navigational queries. In detail, comparing to the global model 44.9% navigational queries get improved MAP

results and only 10.2% of them become worse. "HowTo," "Health" and "Q&A" are the major categories of queries on which our method failed to generate better ranking than the global model. We investigated such kind of queries in the user data set and found the users' clicks are mostly for exploration purposes in these kinds of informational queries (diverse clicked documents), since they might not have a clear mind of answers for these queries yet. As a result, such clicks are less reliable for updating the ranking model.

To understand what types of users can benefit from our adaptation method, we categorized the users in our collection into three classes by the number of adaptation queries they have: 1) *heavy* user, who has more than 10 adaptation queries; 2) *medium* user, who has 5 to 10 adaptation queries; and 3) *light* user, who has less than 5 adaptation queries. We calculated the ranking performance gain from *Cross*-LambdaRank against the global model averaged over the users in these three classes in Table 5. Besides, we also included the improvement from RA-LambdaRank against the global model in the table for comparison.

Table 5: User-level ranking performance gain over global model from *Cross*-LambdaRank and RA-LambdaRank.

Method	User Class	ΔMAP	ΔP@1	ΔP@3	ΔMRR
RA	Heavy	0.1843	0.3309	0.0120	0.1832
	Medium	0.1102	0.2129	0.0025	0.1103
	Light	0.0042	0.0575	-0.0221	0.0041
Cross	Heavy	0.1998	0.3523	0.0182	0.1994
	Medium	0.1494	0.2561	0.0208	0.1500
	Light	0.0403	0.0894	-0.0021	0.0406

All the differences in Table 5, except the *Cross*/Light/ΔP@3 with the value of -0.0021, are significant with p-value<0.01 under paired t-test. We can observe that on the heavy users, who only cover 6.8% population in our collection, *Cross*-LambdaRank and RA-LambdaRank achieved similar improvements against the global model; while on the medium and light users, who consist 14.9% and 78.3% of the whole collection, *Cross*-LambdaRank achieved much more remarkably improvement than RA-LambdaRank. On the heavy users, both methods get relatively sufficient observations from each user to adapt the global model; while on the medium and light users, the observations become scattered and many features are not observed during adaptation. RA-LambdaRank failed to adjust the unseen features properly and only achieved modest improvement over the global model. By sharing transformation across features, *Cross*-LambdaRank better exploited the information conveyed in the limited adaptation data, and obtained better improvement against the global model. Besides, we also found that on the light users, both methods gave degraded P@3 results (the degradation of the *Cross*-LambdaRank is insignificant). We analyzed the results and found that with limited observations in this group of users, the adapted models tend to overfit the original top ranked documents due to positional biases. As a result, the diversity of user preferences might not be properly captured in this type of users.

4.3.3 Adaptation Efficiency

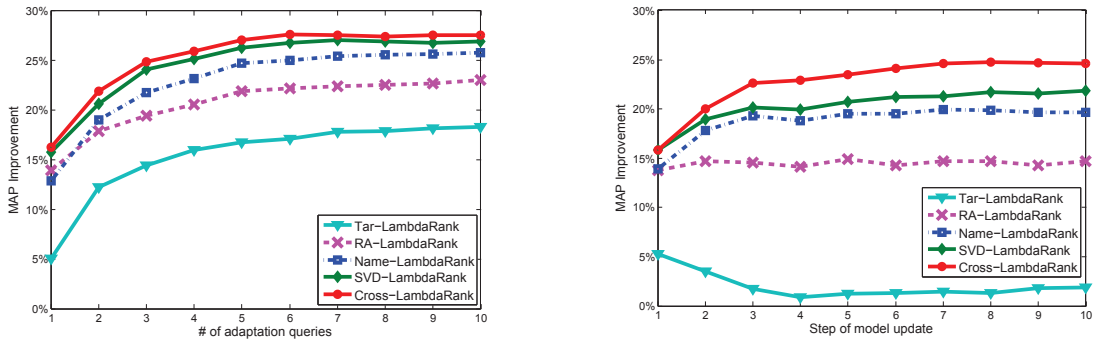
We have discussed different adaptation methods' computation complexity in Section 4.3.1, from which we concluded that the instance-based methods, e.g., TransRank and IW-RankSVM, and feature-based methods, e.g., CLRank, cannot be applied in the per-user basis adaptation scenario, due to the need to frequently access source-domain data during adaptation. Therefore, in this experiment, we will only compare with the model-based adaptation method, i.e., RA method. Besides, we also included the ranking model solely estimated on each user's adaptation data as a baseline. In particular, we will only illustrate the comparison results based on LambdaRank due to its superior ranking performance shown in Section 4.3.2. (Similar results also obtained in RankNet and RankSVM, but due to space limit we cannot include them.) Since all the three methods share similar computational complexity, we evaluated their adaptation efficiency by varying the number of adaptation queries and examining which method can adapt to a user's preferences with fewer number of adaptation queries. To make the results comparable across different settings of adaptation queries, we selected a subset of users who have at least 15 queries in total, in which we fixed the last 5 queries in each user as testing queries. This gives us a collection of 2,743 users with 42,595 queries.

First, we gradually increased the size of adaptation queries from 1 to 10 in each user, and re-estimated the adapted models every time accordingly. The relative improvement of MAP metric for all the methods against the global ranking model on the testing set are shown in Figure 2 (a).

As shown in Figure 2 (a), by leveraging knowledge from the global model, the adapted ranking models outperformed the model only estimated on the adaptation data; and, with only a small amount of adaption data, e.g., 1 or 2 queries, the adapted models can already achieve encouraging improvement (over 15%) against the global model. Comparing with the RA method, our proposed adaptation method achieved more rapid improvement: *Cross*-LambdaRank achieved 25% improvement by using three queries, while RA-LambdaRank slowly climbed to 23% improvement only after 10 queries. Such efficient adaptation credits to the transformation sharing across features in our framework, which helps the model better handle the sparsity problem during adaptation.

The settings in Figure 2 (a) simulate a situation in which batch update is performed, i.e., update the models for each user once we have collected sufficient adaptation data. However, in a more practical setting, we cannot wait too long to collect sufficient adaptation data, so that *online* updating is required. In this experiment, we used the same set of users as in Figure 2 (a), but updated the ranking models for every adaptation query we collected from the user in an online manner, in which we treated the previously updated model as the base model for the next iteration of model updating. The results are shown in Figure 2 (b).

We can clearly notice the advantage of our adaptation method against the baseline methods from the online adaptation results. Because Tar-LambdaRank cannot leverage any knowledge from the global model about the unseen features, its performance fluctuated due to the variance in the adaptation queries. Although RA-LambdaRank appeals to the global model for estimating the unseen features, in the online setting, the knowledge from global model gets diminished rapidly as adaptation evolves, because it only treats the model from last iteration as regularization. As a result, its performance is worse than that in the batch mode (in Figure 2 (a)). In our method, observations in the adaptation data can be fully exploited via transformation sharing, so that unseen features can also be properly updated during online adaptation, which leads to consistent improvement of ranking performance in both batch and online settings.



(a) Batch adaptation for LambdaRank

(b) Online adaptation for LambdaRank

Figure 2: Adaptation efficiency comparisons.

5. CONCLUSIONS

In this work, we proposed a general ranking model adaptation framework for personalized search. A series of learned linear transformations, e.g., scaling and shifting, were performed on the parameters of a generic linear ranking model in a per-user basis, such that the adapted model can better fit each individual user's search result ranking preferences. By sharing transformations across features in a group-wise manner, unseen features can also be properly updated given only limited number of adaptation queries. We instantiated the proposed framework with three frequently used learning-to-rank algorithms, i.e., RankNet, LambdaRank and RankSVM, and the adaptation method achieved significant improvement in, not only adaptation efficiency, but also ranking performance of the adapted ranking models, against several state-of-the-art ranking model adaptation methods in extensive experimentation.

In our current solution, the feature grouping function and transformation matrix are estimated independently. It would be meaningful to jointly estimate the two components for better adaptation performance. Besides, the proposed linear transformation based ranking model adaptation framework opens an interesting new direction for personalization: rich signals, e.g., user-specific profiles and features, could also be included to affect the transformation in order to better reflect users' individual search interests.

6. REFERENCES

[1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR'06*, 2006.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[3] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR'12*, pages 185–194, 2012.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML'05*, pages 89–96. ACM, 2005.

[5] D. Chen, Y. Xiong, J. Yan, G. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253, 2010.

[6] D. Chen, J. Yan, G. Wang, Y. Xiong, W. Fan, and Z. Chen. Transrank: A novel algorithm for transfer of rank learning. In *ICDMW'08*, pages 106–115. IEEE, 2008.

[7] P. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using odp metadata to personalize search. In *SIGIR'05*, pages 178–185. ACM, 2005.

[8] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD'01*, pages 269–274. ACM, 2001.

[9] Z. Dou, R. Song, and J. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW'07*, pages 581–590. ACM, 2007.

[10] K. Duh and K. Kirchhoff. Learning to rank with partially labeled data. In *SIGIR'08*, pages 251–258. ACM, 2008.

[11] R. Fidel and M. Crandall. Users' perception of the performance of a filtering system. In *SIGIR'97*, volume 31, pages 198–205. ACM, 1997.

[12] J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, and H. Zhou. Model adaptation via model interpolation and boosting for web search ranking. In *EMNLP'09*, pages 505–513, 2009.

[13] W. Gao, P. Cai, K. Wong, and A. Zhou. Learning to rank only using training data from related domain. In *SIGIR'10*, pages 162–169. ACM, 2010.

[14] B. Geng, L. Yang, C. Xu, and X. Hua. Ranking model adaptation for domain-specific search. *Knowledge and Data Engineering, IEEE Transactions on*, 24(4):745–758, 2012.

[15] X. He and W. Chou. Minimum classification error linear regression for acoustic model adaptation of continuous density hmms. In *ICME'03*, pages I–397. IEEE, 2003.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *KDD'02*, pages 133–142, 2002.

[17] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR'05*, pages 154–161. ACM, 2005.

[18] C. Leggetter and P. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech and language*, 9(2):171, 1995.

[19] T. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[20] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR'07 workshop on learning to rank for information retrieval*, pages 3–10, 2007.

[21] S. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[22] C. Quoc and V. Le. Learning to rank with nonsmooth cost functions. In *NIPS'07*, volume 19, page 193, 2007.

[23] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR'05*, pages 43–50. ACM, 2005.

[24] D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W. White, S. Dumais, and B. Billerbeck. Probabilistic models for personalizing web search. In *WSDM'12*, pages 433–442, 2012.

[25] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *KDD'06*, pages 718–723, 2006.

[26] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR'05*, pages 449–456. ACM, 2005.

[27] J. Teevan, S. Dumais, and D. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR'08*, pages 163–170. ACM, 2008.

[28] J. Teevan, D. J. Liebling, and G. Ravichandran Geetha. Understanding and predicting personal navigation. In *WSDM'11*, pages 85–94, 2011.

[29] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *WWW'07*, pages 21–30, 2007.