7-2010

Learning to rank only using training data from related domain

Wei GAO
Singapore Management University, weigao@smu.edu.sg

Peng CAI

Kam-Fai WONG

Aoying ZHOU

Learning to Rank Only Using Training Data from Related Domain

Wei Gao[1], Peng Cai[2]*, Kam-Fai Wong[1], and Aoying Zhou[2]

[1]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China
{wgao, kfwong}@se.cuhk.edu.hk
[2]East China Normal University, Shanghai, China
pengcai2010@gmail.com, ayzhou@sei.ecnu.edu.cn

ABSTRACT

Like traditional supervised and semi-supervised algorithms, learning to rank for information retrieval requires document annotations provided by domain experts. It is costly to annotate training data for different search domains and tasks. We propose to exploit training data annotated for a related domain to learn to rank retrieved documents in the target domain, in which no labeled data is available. We present a simple yet effective approach based on instance-weighting scheme. Our method first estimates the importance of each related-domain document relative to the target domain. Then heuristics are studied to transform the importance of individual documents to the pairwise weights of document pairs, which can be directly incorporated into the popular ranking algorithms. Due to importance weighting, ranking model trained on related domain is highly adaptable to the data of target domain. Ranking adaptation experiments on LETOR3.0 dataset [27] demonstrate that with a fair amount of related-domain training data, our method significantly outperforms the baseline without weighting, and most of time is not significantly worse than an "ideal" model directly trained on target domain.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Storage and Retrieval – Retrieval models

General Terms

Algorithms, Experimentation

Keywords

instance weighting, related domain, learning to rank, domain adaptation, RankSVM, RankNet

*This work was done when the author was visiting the Chinese University of Hong Kong

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

1. INTRODUCTION

Ranking in information retrieval (IR) aims to order retrieved documents according to their relevance to a given query. In recent years, learning-based ranking algorithms, known as learning to rank, were extensively studied in IR related communities [5, 6, 15, 18, 35]. Compared to the traditional approaches, such as vector space model [30], BM25 [29] and language-modeling method [26, 37], learning to rank aims to optimize a ranking function that incorporates a wide variety of relevance features and avoid tuning a large number of parameters empirically.

Like supervised algorithms, learning to rank requires large training sets annotated for the specific domain where search tasks are performed, for which much effort should be made by domain experts. For example, building a medical search engine needs experts and labeling standard different from what a musical search engine does; in TREC's Web track [11], named page finding and topic distillation are different tasks, and separate sets of queries and relevance judgements have to be prepared. It is prohibitive to annotate documents for each search domain, especially when fast deployment of a ranking model is demanded.

A promising direction for solving this predicament is to transfer ranking knowledge from the training data of a related domain to the target domain, where only a few or no labeled data is available[1]. Since common information may exist between two domains, we can expect to reuse the training data to derive a ranking model for the target domain considering this kind of commonality. However, due to the different joint distributions of feature and relevance, ranking model directly trained on the related domain generally cannot perform well when tested in target domain.

Cross-domain adaptation has been well studied for classification in nature language processing [4, 12, 21] and machine learning [2, 3, 13]. Although the similar intuition can be applied in ranking adaptation, the fundamental distinctions between ranking and classification need to be considered in algorithm design. In ranking, the main concern is the preference order of two documents or the full order of a list of documents. It is difficult to directly apply classifier adaptation for ranking [14, 17].

Recently, ranking adaptation was received more and more attention [7, 8, 10, 16, 17]. Existing approaches assume a small amount of training data in target domain, which plays an important role in transferring ranking knowledge across domains. However, a small set of labeled data of target domain can only convey limited rank-

[1]Throughout this paper, target domain refers to the domain where the search and ranking is performed, and related domain refers to a domain where large training sets are available for learning a ranking model. Related domain was also called source domain in the literature.

ing knowledge, and other useful information not contained in them can be important but is ignored.

In this paper, we assume a more general scenario where no labeled data but a large number of unlabeled data are available with target domain. We propose to learn a target ranking model by only using related-domain *training data* that are weighted appropriately with the aid of *unlabeled data* in target domain. A simple yet effective instance-weighting framework is created that consists of two steps: (1) we estimate the importance of each related-domain document to the target domain; (2) heuristic schemes are explored to transform the document importance to the weights of document pairs that can be directly incorporated into the popular pairwise ranking algorithms. Due to importance weighting, the ranking function trained on the related-domain data can be highly adaptable to the data in target domain.

The remainder of the paper is organized as follows: Related work are given in Section 2; Section 3 reviews general ranking framework; Section 4 analyzes the problem in ranking across domains; Section 5 presents our importance weighting on documents for ranking adaptation; Experiments and results are discussed in Section 6; Finally, we conclude in Section 7.

2. RELATED WORK

Learning to rank is to optimize a ranking function given data consisting of queries, the retrieved documents and their relevance judgements. Given a new query, the learned function is used to predict the order of retrieved documents. Based on input spaces, three categories of approaches have been proposed, namely pointwise, pairwise and listwise approach. Probabilistic classification and metric regression are typically used in pointwise approach. Popular ranking models like Ranking SVM [18], RankBoost [15], RankNet [5], etc., aim to optimize pairwise loss based on order preference and classify the relevance order between two documents, thus falling into the pairwise approach. Listwise approach [6, 35] considers the entire group of documents associated with the same query in the input space. A comprehensive survey on learning to rank is given in [23].

Domain adaptation was originally to address the classification problems where training data and test data do not follow the same distribution. Existing approaches can be grouped into two directions, namely instance-based and feature-based approaches. The former assumes the same feature space between two domains and attempts to estimate the weights of individual examples in the related domain, which measures their importance to the target domain [19, 21, 36]. The latter tries to infer some common feature representation to bridge the gap between data distributions of two domains, which is regarded as feature weighting scheme [2, 4, 12, 24]. Our method falls into instance-weighting approach.

Existing ranking adaption methods assume a large set of training data in related domain and a small amount of labeled data in target domain. A tree-based adaptation algorithm was proposed by [10]. Gradient Boosting Tree (GBT) was first used to train a ranking model on the related domain. The tree structure was then adjusted with the labeled data in target domain. TransRank [8] proposed to extract k-best adaptive queries in related domain used for training with the help of *labeled* data in target domain. Based on the same assumption of available target-domain labels, [7] presented methods for feature-level as well as instance-level adaptation. [17] treated the model parameters learned from related domain as prior knowledge, and the similarity of parameters between two domains is considered when training on target domain. [16] used model interpolation and error-driven approaches for ranking adaptation, where ranking model was trained on target domain and then combined with a background model learned on the related domain. [34] studied ranking adaptation in heterogeneous domains. [9, 17] proposed ranking adaptability and domain similarity measures. Different from these methods, no labeled data is assumed available in the target domain for our study.

Ranking model training with test data can be thought of as a semi-supervised learning that treats separate domains like the same domain [14, 22]. In these methods, training must be done online once new test data come in, and this is inefficient for web search ranking. In contrast, our training is off-line since test data are not seen at training time.

3. RANKING MODEL REVIEW

Typical training data for learning a ranking model is a set of input-output pairs $(\vec{x}, y)$, where $\vec{x}$ is a feature vector containing relevance scores of a query-document pair, and y is the rank label for the document (e.g., relevance or irrelevant). The features commonly include query-dependent relevance measures such as term frequency and BM25, and query-independent measures such as PageRank. The objective is to optimize a scoring function $f(\vec{x})$ that maps $\vec{x}$ to a real value that indicates the ranking score of the document given the query. If $f(\vec{x_i}) > f(\vec{x_j})$ for documents d_i and d_j, then d_i should be ranked higher than d_j, denoted as $d_i \succ d_j$. Our method is based on pairwise approach, although it can be easily generalized to other approaches.

Pairwise algorithms, such as RankSVM [18] and RankNet [5], aim to minimize a loss based on the preference order of each pair of documents. RankSVM minimizes the number of discordant pairs and maximizes the margin of pair, which is equal to minimize the $L2$ norm of the model f's hyperplane parameter and a Hinge Loss on pairs:

$$\min_{f*} \ \lambda||f||^2 + \sum_{i,j=1}^{\ell} \left(1 - z_{ij} * f(\vec{x_i} - \vec{y_j})\right)^+ \qquad (1)$$

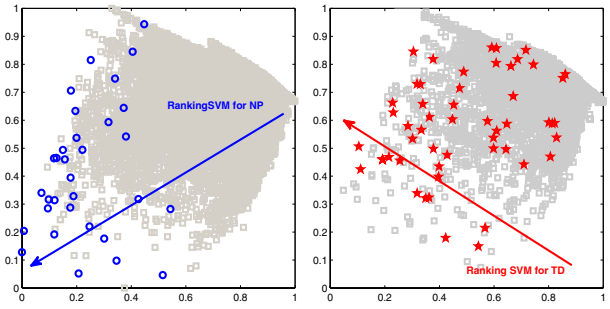
where $z_{ij} = \begin{cases} +1, & \text{if} \quad d_i \succ d_j; \\ -1, & \text{if} \quad d_j \succ d_i \end{cases}$ is the binary preference depending on the ground truth of two documents, $(.)^+$ is the pairwise hinge loss function, λ is the coefficient of trade-off between model complexity and loss term, and ℓ is the number of documents of the query.

At high level, RankNet differs from RankSVM in that it minimizes a loss function based on cross entropy that models the posterior probabilities of rank order [5], which has the following form:

$$\min_{f*} \ \lambda||f||^2 + \sum_{i,j=1}^{\ell} L(P_{ij}, \bar{P}_{ij}) \qquad (2)$$

$L(P_{ij}, \bar{P}_{ij}) \equiv -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$ is the cross entropy loss of a pair (d_i, d_j), where P_{ij} is the posterior $P(d_i \succ d_j)$, and $\bar{P}_{ij}$ is the desired target values for the posteriors and takes one of three values $\{0, 0.5, 1\}$ depending on the ground truth which includes the ties (documents with the same labels). P_{ij} is mapped from outputs of f using a logistic function: $P_{ij} \equiv \frac{e^{o_{ij}}}{1+e^{o_{ij}}}$, where $o_{ij} = f(\vec{x_i}) - f(\vec{x_j})$.

As compared to RankSVM, RankNet usually does not consider kernel and its loss function is pairwise differentiable. Thus, gradient-based optimizer can be used. Also, RankNet is modeled as a two-layer neural network, where input and hidden nodes correspond to features and their combinations. In theory, it can capture any nonlinear correlations among features. RankBoost [15] is another typical pairwise algorithm, for which we do not give the details, as it is similar for our method to apply (see Section 5).



(a) Named page finding (b) Topic distillation

Figure 1: Ranking directions of RankSVM on name page (NP) and topic distillation (TD) queries in TREC-2003.

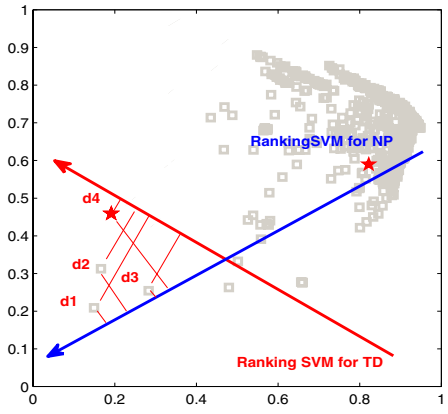


Figure 2: Ranking errors are made for TD query #18 when applying NP model to TD task.

4. PROBLEM ANALYSIS

Document distributions, i.e., the joint distribution of features and relevance of documents, are commonly different between domains. This is caused by many reasons such as different languages, countries and topics [10]. Therefore, ranking model directly trained on one domain may not perform well on the other. Figure 1 and 2 illustrate this predicament of cross-domain ranking using RankSVM on two of TREC's 2003 Web track tasks, namely named page finding (NP) and topic distillation (TD) [11] (Section 6 gives details of the tasks). To plot the figures, we conducted principle component analysis (PCA) on this TREC collection that was released through LETOR3.0 [27] for learning to rank study[2]. Widely used in multivariate analysis, PCA computes a linear combination of features to extract patterns and reduce the dimensionality of data. The goal is to find the direction, known as the principal axis, along which the most variance of the patterns in the data set is captured. The projection of a data point on the principal axis is called the principal component.

Let the horizontal axis and vertical axis represent the first and second principle axes, respectively, and let the blue circles (and red stars) denote relevant documents and the gray squares denote irrelevant ones. The arrowed line in each figure is the ranking direction of RankSVM learned for the corresponding task. As shown in Figure 1(a) and 1(b), documents for NP and TD queries follow different distributions in this two-dimensional PCA space even though they share the same original feature space. Their ranking directions are clearly divergent, which means that when a model trained on one domain is blindly applied to the other domain (moving the two arrowed lines into the same space), projecting documents onto the ranking directions may result in very different orders as compared to the ground truth. This problem is revealed more clearly by Figure 2, where the four documents of TD query #18 are ranked as $d_4 \succ d_2 \succ d_1 \succ d_3$ using the TD model (which is correct as the relevant document d_4 should be placed ahead of others), whereas NP model gives the incorrect order $d_1 \succ d_2 \succ d_3 \succ d_4$. Therefore, the learned NP model can hardly be applied to TD domain directly, and vice versa.

[2]http://research.microsoft.com/en-us/um/beijing/projects/letor/

5. RANKING ADAPTATION BY WEIGHTING IMPORTANCE OF DOCUMENTS

An intuition of ranking adaptation is to tune the ranking direction of related domain to that of target domain as closely as possible. In practice, however, since the data of target domain are not labeled, the target ranking direction cannot be obtained. Our idea is to make use of unlabeled data of target domain for weighting training instances in the related domain so that the knowledge of relevance judgement can be strengthened on those documents that are similar to the target domain. As a result, the learning can be focused on giving better ranking for these important documents. It can be expected that the model trained in this way will work well on ranking actual target-domain documents. To this end, it is critical to measure the extent to which the instances are similar across two domains. The similarity indicates the importance of an instance that can be related to the target domain.

5.1 Cross-Domain Adaptive Ranking Model

Suppose the importance of instances is appropriately weighted in the related domain, we can extend the ranking algorithms described in Section 3 by incorporating the weights into their loss functions. The rationale is that the loss on important instances should be penalized more heavily once ranking errors are made on these documents. Weighting can be developed corresponding to the levels of instance. Pairwise models take document pairs as instances. Thus, the weighting should be made compatible with the pairwise loss.

Let $IW(x_i, x_j)$ be pairwise importance weight for (d_i, d_j), the objective function can be extended for adaptation as follows:

$$\min_{f*} \lambda||f||^2 + \sum_{i,j=1}^{\ell} IW(x_i, x_j) * L_{ij}(.) \quad (3)$$

where $L_{ij}(.)$ is the loss function on pairs that takes the forms like Equations 1 and 2. Note that this scheme can be similarly applied to other pairwise models such as RankBoost. Overall, adaptation training consists of two key steps: (1) instance weighting; (2) weight-based model training.

However, it is difficult to estimate the importance of *pairs* in related domain without labeled data in target domain. This is because document pairs are constructed from preference orders based on ground truth. When rank labels are not provided, one has to enumerate *all* possible pairs. This may produce tremendous noises rendering weight estimation unreliable as most of the pairs are not meaningful for ranking. Weighting individual documents is intuitively more straightforward. Then we can try to derive pairwise weights from document weights.

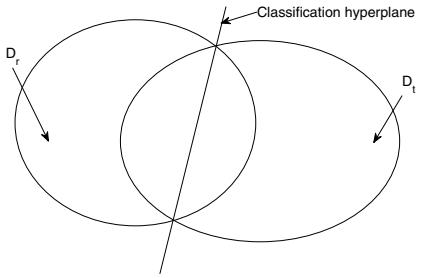


Figure 3: A classification hyperplane can be built between D_r and D_t to measure the extent, to which query-document instance in the related domain is similar to the target domain.

5.2 Importance Weighting Schemes

5.2.1 Weighting Individual Documents

Despite that target-domain data are not labeled, the relevance features of documents for the ranking purpose are readily available to us. These features may not only reflect document's relevancy, but also encode the correlation or similarity of documents in different domains, which conceals some important cross-domain commonalities. For example, many query-independent features that are not focused on query-document relevancy, such as document length, link structure, anchor text, etc., may express some important intrinsic properties of documents. Their discriminative nature is expected to correlate the similar documents across domains and differentiate those that are not similar.

We model document weighting as a classification problem based on the relevance features and the source of domains the document come from. Figure 3 explains this intuition. Given document sets D_r and D_t that denote respectively the related and target domain, we build a classification hyperplane H_{rt} to separate D_r from D_t. If a D_r example is similar to a large number of D_t examples, it should be close to H_{rt}. Hence, we can measure the importance of a related-domain document using its distance to the hyperplane. In practice, the probability that a document in the related domain may be classified into the target domain is adopted.

Algorithm 1 Weight estimation for training documents in related domain

Input:
Related-domain documents, $D_r = \{\vec{x}_{i_r}\}_{i_r=1}^{m}$;
Target-domain documents, $D_t = \{\vec{x}_{i_t}\}_{i_t=1}^{n}$;
Output:
Importance weights of related-domain documents, $IW = \{w_{i_r}\}_{i_r=1}^{m}$;
1: $l_r = +1$, $D'_r = \{(\vec{x}_{i_r}, l_r)\}_{i_r=1}^{m}$;
2: $l_t = -1$, $D'_t = \{(\vec{x}_{i_t}, l_t)\}_{i_t=1}^{n}$;
3: Find classification hyperplane H_{rt} which separates D'_r from D'_t;
4: **for** $i = 1; i \leq m; i++$ **do**
5: Calculate $\ell(\vec{x}_{i_r})$, the distance of $\vec{x}_{i_r}$ to H_{rt};
6: $P(\vec{x}_{i_r} \in D_t|q_r) = \frac{1}{1+exp(\alpha\ell(\vec{x}_{i_r})+\beta)}$;
7: Add $w_{i_r|q_r} = P(\vec{x}_{i_r} \in D_t|q_r)$ to IW;
8: **end for**
9: **return** IW;

Algorithm 1 estimates the probability that each $\vec{x}_{i_r}$ can be classified into D_t, denoted as $P(\vec{x}_{i_r} \in D_t|q_r)$, indicating the importance of $\vec{x}_{i_r}$ given a related-domain query q_r. In step 1-2, the algorithm constructs a training corpus using ± 1, the source of domain, as class label. Then it solves the hyperplane that separates the two domains. For each related-domain document, its distance to the hyperplane is calculated in step 6. A sigmoid function is employed to transform the distance into a posterior probability [25], for which free parameters α and β can be determined by cross-validation or hold-out method. Note that the probability is conditional as each document is in the search result of the given query.

5.2.2 Weighting Document Pairs

Document weights obtained above can be easily integrated into pointwise ranking algorithms. But they cannot be used directly in pairwise algorithms that are generally more effective. Here we address how to transform document weighting to pairwise weighting.

The weight of a pair of documents in related domain can be approximated using the marginal probability of each document in that pair. For a document pair $\langle \vec{x}_{i_r}, \vec{x}_{j_r} \rangle$ of a query q_r from related domain, we can reasonably assume that $P(\vec{x}_{i_r} \in D_t|q_r)$ is independent of $P(\vec{x}_{j_r} \in D_t|q_r)$. Thus, the conditional joint probability $P(\langle \vec{x}_{i_r}, \vec{x}_{j_r} \rangle \in D_t|q_r)$ can be calculated straightforwardly based on this independence assumption:

$$\begin{aligned} w_{i_r,j_r|q_r} &= P(\langle \vec{x}_{i_r}, \vec{x}_{j_r} \rangle \in D_t|q_r) \\ &= P(\vec{x}_{i_r} \in D_t|q_r) \cdot P(\vec{x}_{j_r} \in D_t|q_r) \end{aligned} \quad (4)$$

where $w_{i_r,j_r|q_r}$ denotes the pairwise weight given q_r and can be directly applied to substitute the importance weight in Equation 3.

5.2.3 Weighting Queries

Document weighting and pairwise weighting do not explicitly measure the importance of individual queries. Ranking algorithms constrain the instances to be learned in the range of queries. It is thus reasonable to take into account how likely each query is related to the target domain. If a query is not similar to the type of queries across domains, its retrieved documents should be weighted with lower values accordingly. This aims to generate a model that favors important queries.

We estimate query importance based on the weighting of document pairs considering the compatibility with the pairwise loss. Assuming that the occurrence of document pairs are independent events given a query, query importance can be estimated as the mean value of the importance probabilities of all available pairs, which is formulated as follows:

$$w_{q_r} = P(q_r \in D_t) = \frac{\sum_{i_r,j_r} P(\langle \vec{x}_{i_r}, \vec{x}_{j_r} \rangle \in D_t|q_r)}{M} \quad (5)$$

where M denotes the number of document pairs in query q_r that can be derived from the ground truth.

When applied to pairwise ranking algorithms, Equation 5 can simply substitute the importance weight in Equation 3.

5.2.4 Weight Combination

If we regard ranking across the related and target domains as a two-step procedure, Equations 4 and 5 can be combined in a principled way so that different weighting schemes naturally become complementary to each other since they reflect the degree of cross-domain similarity from different perspectives. The first step is to select an important query that is similar to the target domain, and with this condition, the second step is to choose a pair of important documents from the search result of this query. This process is modeled as follows by the probability $P(\langle q_r, \vec{x}_{i_r}, \vec{x}_{j_r} \rangle \in D_t)$:

$$\begin{aligned} w_{q_r,i_r,j_r} &= P(\langle q_r, \vec{x}_{i_r}, \vec{x}_{j_r} \rangle \in D_t) \\ &= P(q_r \in D_t) \cdot P(\langle \vec{x}_{i_r}, \vec{x}_{j_r} \rangle \in D_t|q_r) \end{aligned} \quad (6)$$

The advantage of the combined weighting scheme above lies in the scaling effect of query weight that influences pairwise weight

Algorithm 2 Weight-based adaptive ranking model training

Input:
Related-domain documents, $D_r = \{\vec{x}_{ir}\}_{ir=1}^{m}$;
Target-domain documents, $D_t = \{\vec{x}_{it}\}_{it=1}^{n}$;
Weighting scheme $K \in \{pair, query, comb\}$
Output:
Ranking model f;
1: $D'_r = \{(\vec{x}_{ir}, +1)\}_{ir=1}^{m}$;
2: $D'_t = \{(\vec{x}_{it}, -1)\}_{it=1}^{n}$;
3: Find classification hyperplane H_{rt} which separates D'_r from D'_t;
4: Split D_r into D_{r1} and D_{r2};
5: Calculate IW_{r1}^{K}, IW_{r2}^{K} according to K;
6: Training model on D_{r1} with weight IW_{r1}^{K} ;
7: Select model f which makes minimum weighted loss on D_{r2} with IW_{r2}^{K};
8: **return** f;

in two aspects: (1) if many document pairs of query q_r have high weight values, then $P(q_r \in D_t)$ should be also high. The scaling tends to strengthen the importance of $\langle \vec{x}_{ir}, \vec{x}_{jr} \rangle$; (2) if many pairs are weighted low and only a few pairs high, the scaling can lower down the importance of these a few pairs that may be outliers.

5.3 The Framework

Algorithm 2 summarizes our overall framework, which contains two key components, i.e., instance weighting and weight-based model training.

Instance weighting is done by steps 1-5. Steps 6 and 7 perform weight-based model training and selection. Note that D_r is split into two parts in step 4, where D_{r1} is used for weight-based training and D_{r2} is for weight-based model selection (validation). With no labeled data in target domain, models can be validated only using labeled data in related domain. In step 7, we select the model f that produces minimum weighted loss on D_{r2}, where the loss is weighted with IW_{r2}^{K}. Including weights makes model selection more meaningful because the best performance on D_{r2} without considering the weights may not generalize well to the target domain. Weight-based validation can select the best model biased towards those highly weighted instances, and thus can adapt better to the target domain.

We use the data in target domain, which are unseen during training, to evaluate this framework. Thus, our method falls into inductive learning [32]. This is an important difference of our method from transductive and semi-supervised adaptation [1, 14] where test data can can be seen during training.

6. EXPERIMENTS AND RESULTS

6.1 Dataset and Setup

We evaluated the proposed framework on TREC-2003 and 2004 Web track datasets, which were released through LETOR3.0 benchmark collection for learning to rank research [27]. Three query tasks were defined on the collection, namely topic distillation (TD), home page finding (HP) and named page finding (NP). TD is to find a list of entry points for good websites that can guide user to page collection which cover the topic. HP is to find the home page of entity such as an person or organization. NP focuses on the page which directly contains the short description of query but not entry point to website [33]. HP and NP tasks are more similar to each other, and TD is distinct from the other two. Ranking adaptation takes place when a task is performed (tested) using the models trained on another task.

There are 64 features for describing a query-document instance (see [27]). The number of queries for different tasks are given in Table 1. Each task has three data sets, i.e., for training, validation and testing, each of which was partitioned into five folds. Since no training data in target domain is assumed in our case, we reused the date sets in the following way while maintaining the fold number unchanged:

Table 1: The number of queries in TREC-2003 and TREC-2004 Web track

Query task	2003	2004
Topic distillation	50	75
Home page finding	150	75
Named page finding	150	75

1. All data (training, validation and test sets) in related domain and partial data (training and validation sets) in target domain were used to build classification hyperplane for instance weighting;

2. Related-domain data were grouped into two parts, where one contained only training set for ranking model training, and the other contained validation and test sets for model selection. Note that model selection can only use labeled data from related domain;

3. The test sets in target domains were always kept unseen during training and only used for model testing. The training and validation sets of target domains were not involved.

Ranking model trained on related domain without instance weighting acted as baseline, denoted as *no-weight*. Three weight-based models, namely *pair-weight*, *query-weight* and *comb-weight*, corresponded to pairwise weighting, query weighting and their combination, respectively. In addition, we randomized document weights and used Equation 4 to generate a random weighting method, denoted as *rand-weight*, to testify the solidity of our probabilistic weighting as compared to the groundless random. Finally, a *target-only* model was trained and tested directly on target domain to provide the corresponding "ideal" performance for reference.

We implemented RankSVM and RankNet using the fast Stochastic Gradient Descent (SGD) optimizer[3] [31]. The ranking results were evaluated by averaging the performance over the five folds with cross validation.

6.2 Results

We carried out HP to NP and NP to TD ranking adaptation tasks. Typically, HP and NP are considered similar domains whereas NP and TD are largely different. Other adaptations such as NP to HP and HP to TD are duplicates of above two and omitted here. Performance was measured by Mean Average Precision (MAP) [28] and Normalized Discounted Cumulative Gain (NDCG) [20] at positions 1,3,5 and 10. Due to space limit, NDCG is only given for RankSVM and similar trend of NDCG was achieved by RankNet.

6.2.1 HP to NP Adaptation in the Same Years

As shown in Table 2 and Figure 4, all our three weighting methods outperform the baseline (*no-weight*) based on MAP and NDCG, and *comb-weight* performs best among three weighting methods. T-test on MAP in Table 2 indicates that all improvements over *no-weight* are statistically significant ($p < 0.03$). This is because the

[3]Source codes will be provided upon request. Due to SGD, our RankSVM may be suboptimal and perform a little worse than SVMLight (`http://svmlight.joachims.org`).

Table 2: MAP results of HP to NP adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *pair-weight* and *comb-weight* and *rand-weight*, respectively (confidence level=95%).

model	RankSVM 2003	RankSVM 2004	RankNet 2003	RankNet 2004
no-weight	0.417	0.596	0.497	0.610
pair-weight	**0.502**†	**0.647**†	**0.532**†	**0.661**†
query-weight	**0.510**†	**0.647**†	**0.534**†	0.645
comb-weight	**0.542**†‡	**0.665**†	**0.562**†‡	**0.679**†
rand-weight	0.431	0.618	0.471	0.619
target-only	**0.660**$^{\sharp}$	**0.681**	**0.672**$^{\sharp}$	**0.669**

Table 3: MAP results of NP to TD adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *pair-weight* and *comb-weight* and *rand-weight*, respectively (confidence level=95%).

model	RankSVM 2003	RankSVM 2004	RankNet 2003	RankNet 2004
no-weight	0.146	0.176	0.196	0.164
pair-weight	**0.194**†	0.171	**0.233**†	0.151
query-weight	**0.194**†	0.175	**0.226**†	0.157
comb-weight	**0.222**†‡	0.179	**0.235**†	0.158
rand-weight	0.145	0.163	0.199	0.168
target-only	**0.235**	0.205	**0.266**	0.180

data distributions of two domains are similar, and our method can easily capture cross-domain similarity by weighting the importance of related-domain documents. Furthermore, *comb-weight* is significantly better than both *pair-weight* and *query-weight* on the 2003 data. On the 2004 data, although not significantly, *comb-weight* still outperforms *pair-weight* 2.78% by RankSVM and 2.72% by RankNet. Thus, weight combination is generally better than than the separate weighting of queries and document pairs.

No obvious difference can be observed between *pair-weight* and *query-weight*. The reason is that only one document is labeled as relevant for each home page query. Thus all the generated pairs contain this document, rendering the values of pairwise weight and query weight close to each other, as the query weight is simply calculated as the mean weight over all document pairs of the query.

All our probabilistic weighting methods can always outperform *random-weight* significantly. This is due to the solid probability ground of our method. Interestingly, *random-weight* sometimes can outperform *no-weight*. The reason is that *no-weight* can be considered as the extreme case where weights are equal to 1, which assumes no difference across domains and all the related-domain documents equally important to the target domain. This is however too strong to be true. Although not stable, *random-weight* could be better by randomly taking into account some extent of uncertainty.

6.2.2 NP to TD Adaptation in the Same Years

Topic distillation is fairly different from named page finding not only on task definition but also on data distribution shown as Figure 1 (see Section 4). Here we examine whether ranking adaptation can be done for such two domains that are not similar. NDCG results of RankSVM are presented in Figure 5. MAP results are given in Table 3 for both algorithms.

The lower MAP scores in Table 3 implies that NP to TD adaptation is generally a harder task as compared to HP to NP. There may be less similar features, but because of this the found classification hyperplane tends to be more discriminative between the two domains. Therefore, we can still expect effective importance weighting.

On the 2003 data, weighting methods demonstrate consistent effectiveness as achieved in HP to NP adaptation. T-test on MAP indicates that *pair-weight* and *query-weight* significantly outperform *no-weight* ($p < 0.02$). Also, *comb-weight* significantly outperforms *pair-weight* as well as *query-weight* ($p < 0.04$). This implies that our adaptation method can work well for these two domains that are not similar.

On the 2004 data, however, weighting is not consistently better than *no-weight* and sometimes performs even worse. T-test does not indicate any statistical significance either. We analyzed the reason: Compared to TREC-2003 where 150 named page queries are provided, the number of named page queries in TREC-2004 is only 75 (see Table 1), rendering less amount of training data. It turns out that important documents become even less. Hence the model overfits to this small number of important instances, thus cannot adapt (generalize) well. We believe it can be improved with more named page queries provided, which is proved in the next section.

Table 4: MAP results of 2003 NP to 2004 TD adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *pair-weight* and *comb-weight* and *rand-weight*, respectively (confidence level=95%).

model	RankSVM	RankNet
no-weight	0.145	0.136
pair-weight	**0.164**†	**0.160**†
query-weight	**0.163**†	**0.157**†
comb-weight	**0.166**†	**0.164**†
rand-weight	0.145	0.144
target-only	**0.205**	**0.180**

Table 5: For NP to TD adaptation, the number of NP training documents in different weight ranges.

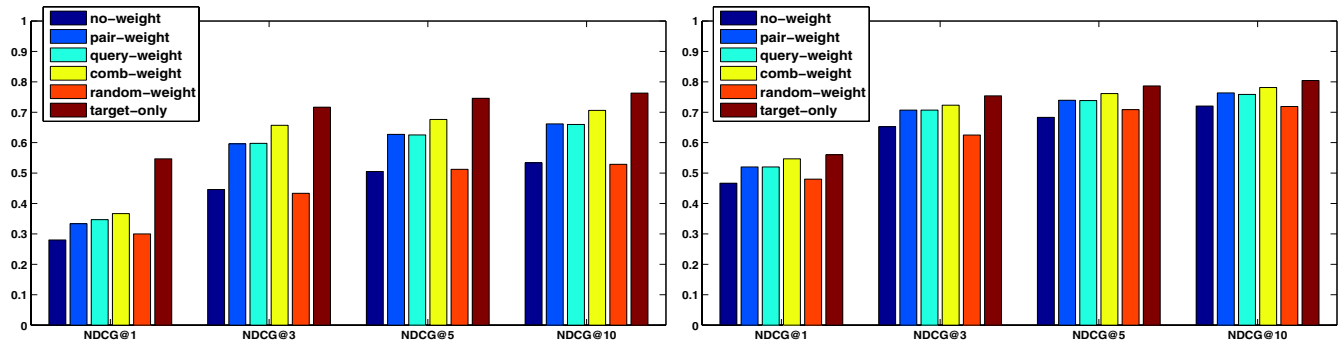
weight range	04 NP→04 TD	03 NP→04 TD
0.5–1	35,773	47,851 (+33.8%)
0.1–0.5	73,968	237,223 (+220.7%)
0–0.1	111,761	160,897 (+44%)

6.2.3 NP 2003 to TD 2004 Adaptation

Here we examine the effectiveness across both different tasks and years. Table 4 shows that weight-based models trained with 2003 NP data significantly outperform *no-weight* when tested on 2004 TD task. This is because a larger number of important documents can be found in NP 2003 which provides 150 name page queries. Table 5 shows statistics on the number of training documents in different ranges of weights we obtained. For example, there are 33.8% more NP 2003 documents than NP 2004 for the importance weight range greater than 0.5 (highly important to target domain). This implies that our weight-based methods favor a fair amount of training data in the related domain for finding enough important documents to avoid overfitting.

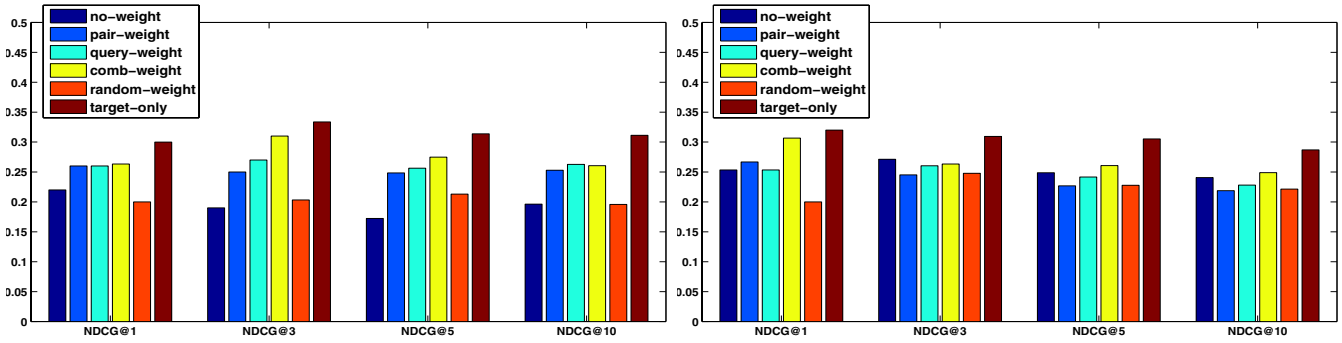
We also studied the influence of weight-based model selection (see Algorithm 2) as compared to the selection without weights. As shown in Table 6, for no-weight selection, weight-based model training can still outperform the baseline, but is consistently worse than using the weight-based selection. This proves that weight-based selection is helpful for adaptation.

We note that RankSVM performs better than RankNet on NP to



(a) HP to NP on TREC-2003 (b) HP to NP on TREC-2004

Figure 4: NDCG of adaptation from home page (HP) finding to named page (NP) finding by RankSVM.



(a) NP to TD on TREC-2003 (b) NP to TD on TREC-2004

Figure 5: NDCG of adaptation from named page (NP) to topic distillation (TD) by RankSVM.

Table 6: Comparison of using and not using weight-based model selection (w. vs n./w.) in 2003 NP to 2004 TD adaptation.

	RankSVM		RankNet	
model	w.	n./w.	w.	n./w.
pair-weight	0.164	0.160	0.160	0.157
query-weight	0.163	0.161	0.157	0.150
comb-weight	0.166	0.164	0.164	0.161
rand-weight	0.145	0.128	0.144	0.130

TD within 2004 and 2003 NP to 2004 TD, but relatively worse on other tasks. This seems to be related to the hyperplane generated by margin-based domain classifier. In easier adaptation, the weights may be less accurate since cross-domain data are more similar and may be not linearly separable. Nonlinear RankNet may generalize better in this case. In difficult adaptation where data are more separable, the weights may be more accurate for RankSVM to take the advantage of maximum margin. We will leave the influences of different weight generation methods on different ranking algorithms for future study.

7. CONCLUSION AND FUTURE WORK

We introduced a simple yet effective instance-weighting framework for ranking adaptation in IR by only using training data from related domain. The basic idea is that the related-domain training instances can be weighted appropriately according to their importance (or similarity) to the target domain. We first estimated the probability of each related-domain document that can be classified into the target domain. Then three instance-weighting methods were proposed based on the document importance. After that, the ranking model was learned in such a way to minimize the weighted loss of ranking instances, so that the ranking function is biased towards those important document and thus can be adaptable to the target domain. Our approach was evaluated on LETOR3.0 dataset for ranking adaptation between different tasks, and we found that it significantly outperformed the baseline without weighting, and most of time was not significantly worse than the model directly trained on the target domain.

Although based on pairwise approach, our method can be generalized easily and combined with other learning to rank approaches. In the future, we will improve the framework from two directions: one is to investigate how different importance estimation methods can influence ranking algorithms; the other is to apply our weighting schemes to various ranking algorithms for ranking adaptation.

8. ACKNOWLEDGEMENT

This work is partially supported by the Innovation and Technology Fund of Hong Kong SAR (No. ITS/182/08) and National 863 program (No. 2009AA01Z150). In addition, P. Cai and A. Zhou are partially supported by an NSFC grant (No. 60925008) and 973 program (No. 2010CB731402). We would like to thank anonymous reviewers for their helpful comments.

9. REFERENCES

[1] M. R. Amini, T. V. Truong, and C. Goutte. A Boosting algorithm for learning bipartite ranking functions with partially labeled data. In *Proceedings of SIGIR*, pages 99–106, 2006.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Proceedings of NIPS*, pages 41–48, 2006.

[3] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[4] J. Blitzer, R. Mcdonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 2006.

[5] 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.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[7] D. Chen, Y. Xiong, J. Yan, G.-R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 2009.

[8] D. Chen, J. Yan, G. Wang, Y. Xiong, W. Fan, and Z. Chen. TransRank: A novel algorithm for transfer of rank learning. In *IEEE International Conference on Data Mining Workshops*, pages 106–115, 2008.

[9] K. Chen, J. Bai, S. Reddy, and B. L. Tseng. On domain similarity and effectiveness of adapting-to-rank. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 1601–1604, 2009.

[10] K. Chen, R. Lu, C. Wong, G. Sun, L. Heck, and B. Tseng. Trada: Tree based ranking function adaptation. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 1143–1152, 2008.

[11] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 web track. In *Proceedings of TREC-2003*, pages 78–92. NIST, 2003.

[12] H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, 2007.

[13] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126, 2006.

[14] K. Duh and K. Kirchhoff. Learning to rank with partially-labeled data. In *Proceedings of SIGIR*, pages 251–258, 2008.

[15] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2004.

[16] 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 *Proceedings of the 2009 Conference on Empirical Methods in Natural Language*, 2009.

[17] B. Geng, L. Yang, C. Xu, and X.-S. Hua. Ranking model adaptation for domain-specific search. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 197–206, 2009.

[18] R. Herbrich, T. Graepel, and K. Obermayer. *Large Margin Rank Boundaries for Ordinal Regression*. MIT Press, Cambridge, 2000.

[19] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Proceedings of NIPS*, pages 601–608, 2006.

[20] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of SIGIR*, pages 41–48, 2000.

[21] J. Jiang and C. Zhai. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, 2007.

[22] M. Li, H. Li, and Z.-H. Zhou. Semi-supervised document retrieval. *Information Processing and Management*, 45(3):341–355, 2009.

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

[24] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, pages 1187–1192, 2009.

[25] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[26] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281, 1998.

[27] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 2010.

[28] S. E. Roberson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[29] S. E. Robertson, S. Walker, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of TREC-3*, pages 109–128, 1995.

[30] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[31] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, 2007.

[32] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

[33] E. M. Voorhees. Overview of TREC 2004. In *Proceedings of TREC-2004*, pages 1–12, 2004.

[34] B. Wang, J. Tang, W. Fan, S. Chen, Z. Yang, and Y. Liu. Heterogeneous cross domain ranking in latent space. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 987–996, 2009.

[35] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of SIGIR*, pages 271–278, 2007.

[36] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the 25th International Conference on Machine Learning*, 2004.

[37] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*, pages 334–342, 2001.