Heterogeneous Cross Domain Ranking in Latent Space

Bo Wang[†*] Jie Tang[‡] Wei Fan[♯] Songcan Chen[†] Zi Yang[‡] Yanzhu Liu[♮*]
[†]Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, China
[‡]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[♯]IBM T.J. Watson Research Center, New York, USA
[♮]Institute of Computer Science and Technology, Peking University, China
{bowang, s.chen}@nuaa.edu.cn, {tangjie, yz}@keg.cs.tsinghua.edu.cn, weifan@us.ibm.com

ABSTRACT

Traditional ranking mainly focuses on one type of data source, and effective modeling still relies on a sufficiently large number of labeled or supervised examples. However, in many real-world applications, in particular with the rapid growth of the Web 2.0, ranking over multiple interrelated (heterogeneous) domains becomes a common situation, where in some domains we may have a large amount of training data while in some other domains we can only collect very little. One important question is: "if there is not sufficient supervision in the domain of interest, how could one borrow labeled information from a related but heterogenous domain to build an accurate model?". This paper explores such an approach by bridging two heterogeneous domains via the latent space. We propose a regularized framework to simultaneously minimize two loss functions corresponding to two related but different information sources, by mapping each domain onto a "shared latent space", capturing similar and transferable concepts. We solve this problem by optimizing the convex upper bound of the non-continuous loss function and derive its generalization bound. Experimental results on three different genres of data sets demonstrate the effectiveness of the proposed approach.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; H.2.8 [**Database applications**]: Data mining

General Terms

Algorithms, Experimentation

Keywords

Heterogeneous cross domain ranking, Transfer ranking, Learning to rank

*This work was done when the first and the last authors are visiting Tsinghua University.

CIKM'09, November 2–6, 2009, Hong Kong, China.

1. INTRODUCTION

Ranking over heterogeneous data sources is an important challenge for many applications. For example, to predict the users' preference (rating score) based on product reviews, one may have much training data (rated reviews) of existing products, but little or no training data for a new product. In social networks, we may have a large amount of training data for movie recommendation, but very limited for recommending friends or web communities. Thus, one basic question is how to make use of the labeled information from existing (source) domain(s) to build an accurate ranking model for the target domain.

Although, quite a few related studies have been conducted, for example, transfer learning [4, 12], domain adaptation [5, 6], multi-task learning [2, 7], learning to rank [9, 15], there are only a few theoretical studies on the heterogeneous cross-domain (HCD) ranking problem. The major difference between the HCD ranking problem and learning to rank is that HCD ranking needs to consider how to borrow the preference order from the source domain (as the supervision information) to the target domain for learning a better ranking model. The HCD ranking problem is also different from transfer learning whose goal is to transfer the knowledge from the source domain to the target domain to learn a classification model. In HCD, the knowledge we desire to transfer is the *preference order* between *heterogeneous* objects from the source domain, instead of their accurate ranking positions.

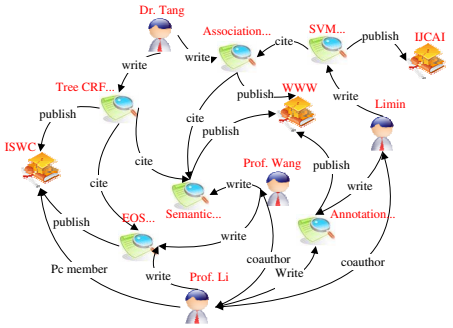
Motivating Application

Figure 1 (a) shows an example of academic search. The objective is to learn functions that can rank different objects for a given query. In Figure 1 (b), the example query is "data mining". The training data (i.e. the labeled rank levels of objects) in some domains, e.g. rank levels of conferences, is relatively easy to obtain (e.g., from several online resources[2]). However, obtaining the training data for some other domains, e.g. papers and authors, would be not obvious. Intuitively, we hope that an approach can take advantage of the available supervision information (labeled conferences) and the correlation between conferences and papers/authors in the academic network to help learn the ranking functions for papers and authors.

Summaries

The challenges of heterogeneous cross-domain (HCD) ranking are as follows:

- *Domain correlation.* As types of objects in the HCD ranking problem may be different (heterogeneous), the



(a) Academic network

(b) Heterogeneous transfer ranking

Figure 1: Example of heterogeneous cross-domain ranking.

first challenge is how to capture the correlation between the source domain and the target domain.

- *Transfer ranking.* It is not only necessary to transfer the knowledge from the source domain to the heterogeneous target domain, but it is also needed to preserve the preference order with the learnt ranking model.

- *Efficiency.* In general, a ranking problem needs thousands of (or millions of) training examples. It is important to develop the method that can scale well to large data sets.

To address the above challenges, we propose a unified cross-domain ranking model, named HCDRank, to simultaneously model the correlation between the source domain and the target domain, as well as learning the ranking functions. In particular, HCDRank uses a "latent feature space" defined over both the source and target domains to measure their correlation. Examples from both domains are mapped onto the new feature space via a projection matrix, where a common (sparse) feature space is discovered. HCDRank adopts a regularization method to simultaneously minimize two loss functions corresponding to the two domains, and supervision from the source domain is transferred to the target domain via the discovered common feature space. An efficient algorithm has been developed and a generalization bound is discussed. Experimental results on three different types of data sets show the proposed approach performs better (+1.2% $\sim$ +6.1% in terms of MAP) than the comparison baseline methods, in particular when the target domain has a very small number of labeled examples. The proposed framework is general, to allow us to utilize many different algorithms to learn the ranking function.

Table 1: List of notations

Notation	Description
$\mathcal{X}_S, \mathcal{X}_T$	instance space for two domains
$\mathcal{Y}_S, \mathcal{Y}_T$	rank level set for two domains
$\mathcal{S}$	unlabeled test set in target domain
$\mathcal{L}_S, \mathcal{L}_T$	labeled data in two domains
p, q	cardinality of rank level set in two domains
n_S, n_T	# queries in two domains
n	# queries in test set
The following are important notations used in Section 3	
f_S, f_T	ranking functions for two domains
w_S, w_T	weight vectors for two domains
α_1, α_2	weight vectors for two domains in latent space
n_1, n_2	# instance pairs in source and target domains
$\|\|W\|\|_{2,1}$	(2,1)-norm of matrix W

2. PROBLEM FORMULATION

The heterogenous cross-domain (HCD) ranking problem can be formalized as follows. For clarity, Table 1 summarizes the notations.

Input: Let $\mathcal{X}_S \in \mathbb{R}^d$ be the instance space of the source domain in which d is the number of features and $\mathcal{Y}_S = \{r_{S_1}, r_{S_2}, \cdots, r_{S_p}\}$ denotes a set of rank levels where p is the number of rank levels in the source domain. The rank levels have: $r_{S_1} \succ r_{S_2} \succ \cdots \succ r_{S_p}$, where $\succ$ denotes the preference relationship. The labeled data in the source domain is denoted by $\mathcal{L}_S = \{(q_S^k, \vec{x}_S^k, \vec{y}_S^k)\}_{k=1}^{n_S}$. That is, for query q_S^k, $\vec{x}_S^k = \{x_{S_i}^k\}_{i=1}^{N_S^k}$ is the related instance collection and $\vec{y}_S^k$ are the corresponding labels where $x_{S_i}^k \in \mathcal{X}_S$, $y_{S_i}^k \in \mathcal{Y}_S$ and N_S^k is the total number of instances related to this query. Further, for the target domain, let $\mathcal{X}_T \in \mathbb{R}^d$ be the instance space and $\mathcal{Y}_T = \{r_{T_1}, r_{T_2}, \cdots, r_{T_q}\}$ is the set of rank levels. There are two parts of data in the target domain: $\mathcal{S} = \{(q^k, \vec{x}^k)\}_{k=1}^{n}$ represents the unlabeled test data in which $x_i^k \in \mathcal{X}_T$ and $\mathcal{L}_T = \{(q_T^k, \vec{x}_T^k, \vec{y}_T^k)\}_{k=1}^{n_T}$ represents the labeled data where $x_{T_i}^k \in \mathcal{X}_T$ and $y_{T_i}^k \in \mathcal{Y}_T$. Note that the labeled data $\mathcal{L}_T$ is not necessary, which implies we may have no labeled target domain data.

Learning task: In the HCD ranking problem, the transfer ranking task can be defined as: given limited number of labeled data $\mathcal{L}_T$, a large number of unlabeled data $\mathcal{S}$ from the target domain, and sufficiently labeled data $\mathcal{L}_S$ from the source domain, the goal is to learn a ranking function f_T^* for predicting the rank levels of unlabeled data in the target domain.

There are several key issues: (1) the source and the target domains may have different feature distributions or different feature spaces (e.g., different types of objects); (2) the number of rank levels in the two domains can be different; (3) the number of labeled training examples in different domains may be very unbalanced (e.g., a thousand vs a few).

3. HCD RANKING

3.1 Basic Idea

In HCD ranking, we aim at transferring preference information from an interrelated (heterogeneous) source domain to the target domain. As the feature distributions and the objects' types may be different across domains, the first chal-

lenge we need to address is how to quantitatively measure the correlation between the different domains, which reflects what kind of information can be transferred across the domains. On the other hand, our ultimate goal is to obtain a higher ranking performance. Based on these considerations, we have two main ideas: first we assume there is a common (latent) space between the two domains. Examples (e.g., x) from the two domains can be mapped onto the latent space through a transformation function $\phi(x)$. Such a common latent space provides a potential way to quantify the correlation between the two domains. Second, in the target domain we aim to learn a ranking model that can minimize the error (loss) on the unlabeled test data while preserving the preference orders in the labeled training data. When transferring the supervision information from the source domain, we also desire to preserve its original preference order, equivalently minimizing the loss in the source domain. Therefore, we propose a general framework (HCDRank), in which we use a latent space to bridge the two domains (i.e., the source domain and the target domain) and define two loss functions respectively for the two domains. We further propose an efficient algorithm to optimize the two loss functions and learn the latent space simultaneously.

3.2 The General Framework: HCDRank

Given the labeled training data from the target domain $\mathcal{L}_T = \{(q_T^k, \vec{x}_T^k, \vec{y}_T^k)\}_{k=1}^{n_T}$, we aim to learn a ranking function f_T which can correctly predict the preference relationships between instances for each query q_T^k, i.e. $f_T(x_{T_i}^k) > f_T(x_{T_j}^k) : \forall y_{T_i}^k \succ y_{T_j}^k$. For ranking, based on the learnt ranking function f_T, we can predict the rank level of a new instance. To learn the ranking function, we can consider to minimize the following loss function:

$$\begin{aligned}\min_{f_T} \mathcal{O}(f_T, \mathcal{L}_T) &= R(f_T, \mathcal{L}_T) + \eta \mathcal{E}(f_T) \\ &= \sum_{k=1}^{n_T} \sum_{y_{T_i}^k \prec y_{T_j}^k} \mathbb{I}[f_T(x_{T_i}^k) > f_T(x_{T_j}^k)] + \eta \mathcal{E}(f_T) \end{aligned} \quad (1)$$

where $\mathbb{I}[\pi]$ is the indicator function returning 1 when π is true and 0 otherwise; $R(f_T, \mathcal{L}_T)$ counts the number of mis-ranked pairs in the target domain; η is a parameter that controls the tradeoff between the empirical loss (the first term R) and the penalty $\mathcal{E}$ (the second term) of the model complexity.

When transferring the supervision information from the source domain, we hope to preserve the preference order between instances from the source domain. For bridging instances from the two heterogeneous domains, we define a transformation function $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ to map instances from both domains to a d'-dimensional common latent space. Then we can define a general objective function for the HCD ranking problem as follows:

$$\begin{aligned}&\min_{f_S, f_T, \phi} R_\phi(f_S, \mathcal{L}_S) + C R_\phi(f_T, \mathcal{L}_T) + \lambda J_\phi(f_S, f_T) \\ &= \sum_{k=1}^{n_S} \sum_{y_{S_i}^k \prec y_{S_j}^k} \mathbb{I}[f_S(\phi(x_{S_i}^k)) > f_S(\phi(x_{S_j}^k))] \\ &+ C \sum_{k=1}^{n_T} \sum_{y_{T_i}^k \prec y_{T_j}^k} \mathbb{I}[f_T(\phi(x_{T_i}^k)) > f_T(\phi(x_{T_j}^k))] \\ &+ \lambda J_\phi(f_S, f_T)\end{aligned} \quad (2)$$

where $J_\phi(f_S, f_T)$ is a penalty for the complexity of the HCD ranking model, λ is a tuning parameter that balances the empirical losses and the penalty, and C is a parameter to control the imbalance of labeled instances between the two domains.

The problem now is to find the best parameters for f_S, f_T and ϕ, that minimize the objective function (Eq. 2). In the following section, we give an instantiation of the framework and present a preferred solution.

3.3 The Proposed Solution

In HCDRank, we do not simply want to learn the ranking function f_T, f_S for the two domains but also learn the transformation function ϕ. In addition, it is desirable to leave out features that are not important for transferring knowledge across domains and result in a sparse solution.

Instantiation of the HCDRank framework. Without loss of generality, f_T is assumed to be a linear function in the instance space: $f_T(x) = \langle w_T, x \rangle$, where w_T are parameters (feature weights) to be estimated from the training data and $\langle \cdot \rangle$ indicates the inner product. By substituting it into Eq. 1 we have

$$\mathcal{O}(f_T, \mathcal{L}_T) = \sum_{k=1}^{n_T} \sum_{y_{T_i}^k \prec y_{T_j}^k} \mathbb{I}[\langle w_T, (x_{T_i}^k - x_{T_j}^k) \rangle > 0] + \eta \mathcal{E}(f_T) \quad (3)$$

The loss function $R(f_T, \mathcal{L}_T)$ is not continuous, so we just use Ranking SVM hinge loss to upper bound the number of mis-ranked pairs [8]. For easy explanation, we define the following notations: for each query $\vec{q}_T^k (k = 1, \cdots, n_T)$, given an instance pair $x_{T_i}^a, x_{T_i}^b$ from different rank levels and their corresponding labels $y_{T_i}^a$, $y_{T_i}^b$, we create a new instance

$$\left(x_{T_i}^a - x_{T_i}^b, \quad z_{T_i} = \begin{cases} +1 & y_{T_i}^a \succ y_{T_i}^b \\ -1 & y_{T_i}^a \prec y_{T_i}^b \end{cases} \right) \quad (4)$$

Then we can get a new training data set consisting of instance pairs in the target domain $\mathcal{L}'_T = \{(x_{T_i}^a - x_{T_i}^b, z_{T_i})\}_{i=1}^{n_2}$. For the source domain, we can make the same assumption and use the parallel notations w_S and $\mathcal{L}'_S = \{(x_{S_i}^a - x_{S_i}^b, z_{S_i})\}_{i=1}^{n_1}$. Finally, we can rewrite the objective function by optimizing the convex upper bound of the original loss as:

$$\begin{aligned}\min_{w_S, w_T, \phi} &\sum_{i=1}^{n_1} [1 - z_{S_i} \langle w_S, (\phi(x_{S_i}^a) - \phi(x_{S_i}^b)) \rangle]_+ \\ &+ C \sum_{i=1}^{n_2} [1 - z_{T_i} \langle w_T, (\phi(x_{T_i}^a) - \phi(x_{T_i}^b)) \rangle]_+ \\ &+ \lambda J_\phi(w_S, w_T)\end{aligned} \quad (5)$$

Now the problem is to define the transformation function and the penalty of the model complexity.

Instantiation of the transformation function and the penalty. We use a $d \times d$ matrix U to describe the correlation between features. The inner product of examples are then defined as $x_i^\top U U^\top x_j$ using the matrix. Such parameterization is equivalent to projecting every example x onto a latent space spanned by $\phi : x \to U^\top x$. With the transformation function, we can redefine the loss function, for example, by replacing the first term in Eq. 5 with:

$$\sum_{i=1}^{n_1} [1 - z_{S_i} \langle w_S, U^\top (x_{S_i}^a - x_{S_i}^b) \rangle]_+ \quad (6)$$

As for the penalty $J_\phi(w_S, w_T)$ of the model complexity, we define it as a regularization term, specifically, a (2,1)-norm $\|W\|_{2,1}$, for the parameters of the source and the target

domains, where $W = [w_S, w_T]$ is a $d \times 2$ matrix with the first column corresponding to w_S and the second w_T. The $(2,1)$-norm of W is defined as $||W||_{2,1} = \sum_{i=1}^{d} ||a^i||_2$ where a^i is the i-th row of W. The 2-norm regularizer on each row of W leads to a common feature set over the two domains and the 1-norm regularizer leads to a sparse solution. The (2,1)-norm regularizer thus offers a principled way to interpret the correlation between the two domains and also introduce useful sparsity effects. Finally, we can redefine the objective function as:

$$\begin{aligned} \min_{w_S, w_T, U} & \sum_{i=1}^{n_1} [1 - z_{S_i} \langle w_S, U^\top (x^a_{S_i} - x^b_{S_i}) \rangle]_+ \\ & + C \sum_{i=1}^{n_2} [1 - z_{T_i} \langle w_T, U^\top (x^a_{T_i} - x^b_{T_i}) \rangle]_+ + \lambda \|W\|^2_{2,1} \\ s.t. \ & U^\top U = I \end{aligned} \tag{7}$$

where $U^\top U = I$ denotes an orthogonal constraint which makes the projection matrix U unique.

Learning algorithm. Directly solving the objective function (involving solving parameters w_S, w_T, U in Eq. 7) is intractable, as it is a non-convex problem. Fortunately, we can derive an equivalently convex formulation of the objective function Eq. 7 as follows: (Derivation of the equivalence is given in the appendix.)

$$\begin{aligned} \min_{M, D} & \sum_{i=1}^{n_1} [1 - z_{S_i} \langle \alpha_1, x^a_{S_i} - x^b_{S_i} \rangle]_+ \\ & + C \sum_{i=1}^{n_2} [1 - z_{T_i} \langle \alpha_2, x^a_{T_i} - x^b_{T_i} \rangle]_+ + \lambda \sum_{t=1}^{2} \langle \alpha_t, D^+ \alpha_t \rangle \\ s.t. \ & D \succeq 0 \\ & \text{trace}(D) \leq 1 \\ & \text{range}(M) \subseteq \text{range}(D) \end{aligned} \tag{8}$$

where $M = [\alpha_1, \alpha_2] = UW$, $D = U \text{Diag}(\frac{||a^i||_2}{||W||_{2,1}}) U^\top$ and the superscript "+" of D indicates the pseudoinverse of the matrix D. X is a $p \times q$ matrix, range of X is the span of columns of X which can be defined as $\text{range}(X) = \{x | Xz = x, \text{for some } z \in R^q\}$. The trace constraint of D is imposed because if D is set to ∞, the objective function will degenerate to only minimize the empirical loss. The range constraint bounds the penalty term below and away from zero. The equivalence has been previously used for multi-task feature learning [2].

We can solve the equivalently convex problem with an iterative minimization algorithm, as outlined in Algorithm 1, and detailed as follows:

Step 1. We use an iterative algorithm to optimize matrix M and D. First, in lines 2-4, we keep D fixed, and learn α_1 and α_2 (that is, matrix M) from the labeled training data in two domains respectively. Second, in line 5, we update matrix D by the learnt matrix M. We run the above two steps iteratively until convergence or excess of the maximal iteration number. Then in lines 7 and 8, we apply SVD decomposition [26] on the learnt intermedia matrix D, i.e. $D = U \Sigma V^\top$; then the matrix U is constructed by the eigenvectors corresponding to the first and second biggest eigenvalues of D.

Step 2. In line 9, we learn the weight vector of the target domain from all the labeled data of two domains in the latent space. In lines 10-12, we use the learnt w^*_T to predict ranking levels of new instances from the target domain.

Complexity. The size of the two matrices to be optimized in HCDRank depends only on the feature number d, e.g., matrix D is $d \times d$ and W is $d \times 2$, and the complexity for SVD decomposition on matrix D is $O(d^3)$.

Let $N = n_1 + n_2$ be the total number of instance pairs for training and s be the number of non-zero features. Using the cutting-plane algorithm[20], linear Ranking SVM training has $O(sN \log(N))$ time complexity. In our algorithm HCDRank, let T be the maximal iteration number, then the training of HCDRank has $O((2T+1) \cdot sN \log(N) + d^3)$ time complexity.

Algorithm 1: HCDRank for transfer ranking

Input: Training set: $\mathcal{L}_S \bigcup \mathcal{L}_T$; Test set: $\mathcal{S}$
Output: Ranking function $f^*_T = \langle w^*_T, x \rangle$ and the predicted preferences over test data: $\{y_i\}_{i=1}^n$
Initialization: $D = \frac{I_{d \times d}}{d}$

Step 1: Latent Space Finding
1: **while** not reached maximal iteration number T **do**
2: $\alpha_1 = \text{argmin}\{\sum_{i=1}^{n_1} [1 - z_{S_i} \langle \alpha, x^a_{S_i} - x^b_{S_i} \rangle]_+ + \lambda \langle \alpha, D^+ \alpha \rangle\}$
3: $\alpha_2 = \text{argmin}\{\sum_{i=1}^{n_2} [1 - z_{T_i} \langle \alpha, x^a_{T_i} - x^b_{T_i} \rangle]_+ + \lambda \langle \alpha, D^+ \alpha \rangle\}$
4: $M = [\alpha_1, \alpha_2]$
5: set $D = \frac{(MM^\top)^{\frac{1}{2}}}{\text{trace}(MM^\top)^{\frac{1}{2}}}$
6: **end while**
7: Apply SVD decomposition on D, $D = U \Sigma V^\top$
8: Construct U by the eigenvectors corresponding to the first and second biggest eigenvalues of D

Step 2: Learning in Latent Space
9: $w^*_T = \text{argmin}\{\sum_{i=1}^{n_1} [1 - z_{S_i} \langle w, U^\top (x^a_{S_i} - x^b_{S_i}) \rangle]_+ + C \sum_{i=1}^{n_2} [1 - z_{T_i} \langle w, U^\top (x^a_{T_i} - x^b_{T_i}) \rangle]_+ + \lambda ||w||^2\}$
10: **for** i = 1 to n **do**
11: $y_i = \langle w^*_T, U^\top x_i \rangle$
12: **end for**

3.4 Generalization Bound

First, let a domain be defined by two terms: the distribution $\mathcal{D}$ on instance space $\mathcal{X}$, and a ranking function $f : \mathcal{X} \to \{r_1, r_2, \cdots, r_p\}$. Then source and target domains are denoted by $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ respectively. Let $\epsilon_S(h)$ and $\epsilon_T(h)$ denote the source and target risks. Correspondingly, $\hat{\epsilon}_S(h)$ and $\hat{\epsilon}_T(h)$ are the empirical risks.

An equivalent formulation for Eq. 7 is as follows:

$$\begin{aligned} \min_{w_S, w_T, U} & \sum_{i=1}^{n_1} [1 - z_{S_i} \langle w_S, U^\top (x^a_{S_i} - x^b_{S_i}) \rangle]_+ \\ & + C \sum_{i=1}^{n_2} [1 - z_{T_i} \langle w_T, U^\top (x^a_{T_i} - x^b_{T_i}) \rangle]_+ \\ s.t. \ & ||W||_{2,1} \leq z \\ & U^\top U = I \end{aligned} \tag{9}$$

where $z \geq 0$ and there is a one-to-one correspondence between λ and z [22].

In Eq. 9, the objective function is $\hat{\epsilon}_S(h) + C \hat{\epsilon}_T(h)$ with parameter $C \in [0, \infty)$. It is easy to prove that C is equivalent to the ratio $\frac{1-\theta}{\theta}$ with $\theta \in [0, 1]$; that is, $\theta = \frac{1}{1+C}$. Thus, by replacing C with $\frac{1-\theta}{\theta}$ and multiplying both sides of the equation by θ, we can obtain the following equivalent objective function which is a convex combination of empirical source and target risk:

$$\hat{\epsilon}_\theta(h) = \theta \hat{\epsilon}_T(h) + (1 - \theta) \hat{\epsilon}_S(h) \tag{10}$$

where $\theta = \frac{1}{1+C}$. $\hat{\epsilon}_\theta(h)$ and $\epsilon_\theta(h)$ are the empirical and true weighted risk respectively. Hereafter, we will analyze the objective function in the formulation of Eq. 10.

THEOREM 1. *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d. Let $\mathcal{U}_S$ and $\mathcal{U}_T$ be unlabeled samples of size m' each, drawn from $\mathcal{D}_S$ and $\mathcal{D}_T$ respectively, and $\hat{d}_{\mathcal{H}\triangle\mathcal{H}}$ is the empirical distance between them. Let $\mathcal{L} = \mathcal{L}_S \bigcup \mathcal{L}_T$ be the labeled samples of size m generated by drawing $(1-\beta)m$ points from $\mathcal{D}_S$ and βm points from $\mathcal{D}_T$, labeling them according to f_S and f_T respectively. For each ranking function h with zero training risk, if $\hat{h} \in \mathcal{H}$ is the empirical minimizer of $\hat{\epsilon}_\theta(h)$ on $\mathcal{L}$, then with probability of at least $1-\delta$(over the choice of the samples)[5, 15]*

$$\epsilon_T(\hat{h}) < \frac{2}{\beta m-1}\left(d\log\left(\frac{8e(\beta m-1)}{d}\right)\log(32(\beta m-1)) + \log\left(\frac{8(\beta m-1)}{\delta}\right)\right) + 2\sqrt{\frac{\theta^2}{\beta} + \frac{(1-\theta)^2}{1-\beta}}\sqrt{\frac{d\log(2m)-\log\delta}{2m}} + 2(1-\theta)\left(\frac{1}{2}\hat{d}_{\mathcal{H}\triangle\mathcal{H}}(\mathcal{U}_S,\mathcal{U}_T) + 4\sqrt{\frac{2d\log(2m')+\log\left(\frac{4}{\delta}\right)}{m'}} + \gamma\right)$$

where $\gamma = \min_{h\in\mathcal{H}} \epsilon_S(h) + \epsilon_T(h)$ and $\beta = \frac{n_T}{n_S+n_T}$.

The error bound is comprised of three components: the first one is the upper bound for the target risk using only the labeled data in the target domain; the second one corresponds to the difference between the true and empirical weighted risks; the last one measures the distance between target risk and weighted risk. Due to space limitation, details of the proof are given in the extended paper.

4. EXPERIMENTS

Our approach is general and can be applied to various data sets. We perform our experiments on three different genres of data sets: a homogeneous data set which consists of documents from different domains; a heterogeneous data set which consists of three different types of objects; a heterogeneous task data set which consists of two different ranking tasks.

4.1 Evaluation Measures, Baseline Methods

Evaluation measures. To quantitatively evaluate our method, we use P@n(Precision@n), MAP (mean average precision)[3] and NDCG (normalized discount cumulative gain) [17].

The precision of top n results for a query is measured by precision at n which is defined as follows:

$$P@n = \frac{\#\{\text{relevant documents in top } n \text{ results}\}}{n}$$

Average precision is defined based on the P@n to measure the accuracy of ranking results for a given query.

$$AP = \sum_n \frac{P@n \cdot \mathbb{I}[\text{document } n \text{ is relevant}]}{\#\{\text{relevant documents}\}}$$

MAP is then defined as the mean of all APs over test set and measures the mean precision of ranking results over all the queries. Different from MAP, NDCG gives high weights to the top ranked relevant documents. The NDCG score at position n is defined as follows:

$$N@n = Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{\log(1+j)}$$

where $r(j)$ is the rank of j-th document, and Z_n is a normalization factor.

Baseline methods. We compare the proposed ranking model HCDRank with three methods as listed in Table 2. Ranking SVM (RSVM) [15] is one of the state-of-the-art ranking algorithms for information retrieval. It is designed for ranking in one domain only. For fair comparison, we conduct two experiments with RSVM, one is to train the ranking model on the target domain $\mathcal{L}_T$ only and the other (called RSVMt) is to train the ranking model by combining the source domain and the target domain $\mathcal{L}_S \bigcup \mathcal{L}_T$. The third comparison method is MTRSVM which is a multi-task feature learning approach using ranking SVM hinge loss[2].

Table 2: Three baseline methods.

	RSVM	RSVMt	MTRSVM	**HCDRank**
Training data	$\mathcal{L}_T$	$\mathcal{L}_S \bigcup \mathcal{L}_T$	$\mathcal{L}_S \bigcup \mathcal{L}_T$	$\mathcal{L}_S \bigcup \mathcal{L}_T$
Test data	$\mathcal{S}$	$\mathcal{S}$	$\mathcal{S}$	$\mathcal{S}$

All the experiments are carried out on a PC running Windows XP with Dual-Core AMD Athlon 64 X2 Processor(2 GHz) and 2 G RAM. We use SVM^{light} [19] with linear kernel and default parameters to implement RSVM, RSVMt and the preference learning step of MTRSVM. The proposed ranking model HCDRank has been implemented using Matlab 7.1 and the maximal iteration number T is set to five. Also without special specification, we use the grid search to choose parameter C from $\{2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 1, 2, 2^2, 2^3, 2^4, 2^5\}$ and the results reported in this paper are all averaged over 10 runs.

4.2 Results on Homogeneous Data

Data Set. We use LETOR 2.0 [23] as the homogeneous data set, which is a data set for evaluating various algorithms for learning to rank. LETOR 2.0 is comprised of three sub data sets: TREC2003, TREC2004, and OHSUMED, with respectively 50, 75, and 106 queries. A set of query-document pairs are collected in each of the data sets. The TREC data is a collection from a topic distillation task which aims to find good entry points principally devoted to a given topic. The OHSUMED data is a collection of records from medical journals. In the OHSUMED data set, there are three rank levels, i.e. relevant $\succ$ partially relevant $\succ$ non-relevant, while in the TREC data set, there are two, i.e. relevant $\succ$ non-relevant. In LETOR, all the features are highly abstract. In TREC, there are 44 features divided into four categories. In OHSUMED, there are 25 features falling into three categories. Table 3 summarizes the features in the LETOR data set. For example, for TREC data, there are 16 low-level content features (e.g. tf and idf), 13 high-level content features (e.g. BM25 and language model for IR), 7 hyperlink features (e.g. PageRank and HITS) and 8 hybrid features (e.g. hyperlink-based relevance propagation).

Feature definition. To adapt to the cross-domain ranking scenario, we make slight revision to the LETOR data set. After revision, the whole data set and three sub data sets are correspondingly referred to as LETOR_TR, TREC2003_TR, TREC2004_TR and OHSUMED_TR. Specifically, we split each data set into two domains (source domain and target domain), according to the feature types. Table 4 lists statistics of the data sets in which the 3th column shows the details for features used in each domains of every data set by feature categories A-H in Table 3. We split features in this way in order to simulate some real applications. For example, the source domain of TREC2003_TR only contains feature categories A and B with queries 1-25 which correspond to features for document contents; while the target domain of TREC2003_TR consists of feature categories B,

Table 3: Description of features in LETOR data set.

Data set	Feature Categories	Description	#Features	Feature IDs
TREC	A	low-level content features	16	{2-5,9-12,28-35}
	B	part of high-level content features	6	{15,16,19,20,23,24}
	C	part of high-level content features	7	{1,17,18,21,22,25,26}
	D	hyperlink features	7	{6-8,14,36-38}
	E	hybrid features	8	{13,27,39-44}
OHSUMED	F	low-level content features for title	10	{1-10}
	G	low-level content features for abstract	10	{11-20}
	H	high-level content features	5	{21-25}

Table 4: Data characteristics of LETOR_TR data set. #D/Q and #Dp/Q respectively denotes the average number of documents and the average number of document-pairs corresponding to a query.

Data Set	Query IDs	Features	#Doc	#D/Q	#Dp/Q
TREC2003_TR					
SOURCE	25:{1-25}	AB	24079	963	6450
TARGET	25:{26-50}	BCDE	25092	1004	13761
TREC2004_TR					
SOURCE	38:{1-107}	AE	37154	978	5969
TARGET	37:{111-221}	BCDE	37016	1000	5696
OHSUMED_TR					
SOURCE	56:{1-56}	FH	8136	145	5726
TARGET	50:{57-106}	GH	8004	160	5239

C, D, E with queries 26-50 which may correspond to features in blogs. After this splitting, intuitively the features in two domains are quite different. In all experiments, we use the labeled related documents for queries from the source domain as the training data $\mathcal{L}_S$, and randomly sample 20% of the queries and the related documents from the target domain as the training data $\mathcal{L}_T$ of the target domain (that is, 5, 8 and 10 queries for TREC2003_TR, TREC2004_TR and OHSUMED_TR respectively), while all the other data in the target domain are viewed as the unlabeled test set $\mathcal{S}$.

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED). We set the values of features that are not defined in a domain as zero. For example, in the source domain of TREC2003_TR, only features of categories A and B are set with their actual values, the values of others (B, C, D, E) are set to zero. Similarly, in the target domain of TREC2003_TR, only features of categories B, C, D and E have their actual values and the others are set to zero.

Results and analysis. Figure 2 and Table 6 show the results of the proposed and the comparison methods on the LETOR_TR data sets. Generally, our approach achieves a higher performance and has a nice convergence property (converging after several iterations in most experiments). Specifically, we have the following observations:

1. **Ranking accuracy.** HCDRank performs much better (by +5.6% and +6.1% respectively in terms of MAP) than the comparison methods on both TREC2003_TR and OHSUMED_TR. On TREC2004_TR, HCDRank results in a comparable performance with RSVMt.

2. **Effect of difference.** We measure the difference of the source domain and the target domain in each data set by the cosine-based similarity. The cosine similarities of the three sub data sets are 0.01, 0.23, and 0.18. We see that when the similarity is relatively high (0.23 on TREC2004_TR), simply combination of the training data from both domains for learning would result in a better ranking performance: RSVMt performs better than MTRSVM and RSVM. When the similarities are relatively low (0.01 on TREC2003_TR and 0.18 on OHSUMED_TR), such a brute combination will introduce a lot of noise which hurts the performance: RSVMt underperforms MTRSVM and RSVM. In both situations, our approach can balance the difference and consistently outperform the three methods.

3. **Reason for performance.** We conduct an analysis of why HCDRank is effective on LETOR_TR. An important observation is that, in the ranking problem, many features are extracted from query-document pairs, that is, the features already contain information from both queries and documents. Thus a good common latent space means that if the new feature representation in that space of query-document pair q_1-d_1 from the source domain is similar to that of query-document pair q_2-d_2 from the target domain, then the rank level of the two documents are also similar with each other. For example, if d_1 is relevant to q_1, then it is highly possible that d_2 is also relevant to q_2.

4. **Training time.** Finally, we compare the training time of different approaches on the three data sets (listed in Table 5). Generally, HCDRank needs relatively more time in the training process. But we need note that the proposed learning algorithm for HCDRank can be easily parallelized (as has been done by [11]) and we need only run the training process once on a data set.

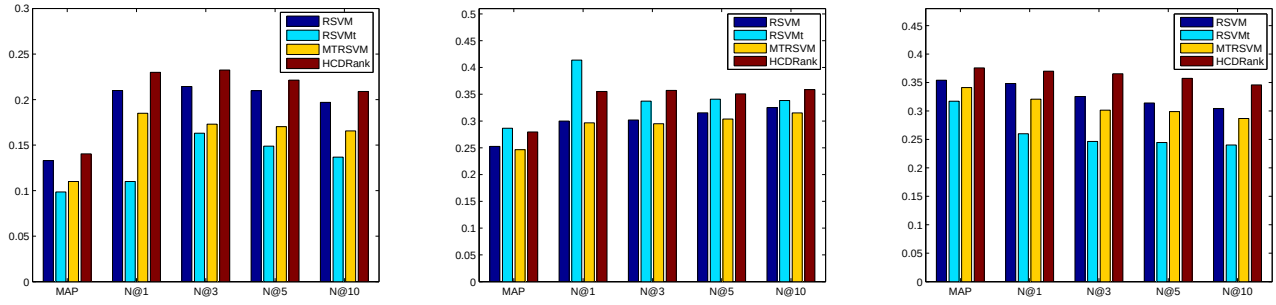
Table 5: Training time used on LETOR_TR (S).

	RSVM	RSVMt	MTRSVM	HCDRank
TREC2003_TR	744	973	6180	6417
TREC2004_TR	83	1125	6810	7548
OHSUMED_TR	647	820	7146	7477

4.3 Results on Heterogeneous Data

Data Set. The second data set is a heterogeneous academic data set, which contains 14,134 authors, 10,716 papers, and 1,434 conferences. The queries are 44 most frequent queried keywords (e.g., "data mining", "information retrieval") collected from the query log of the ArnetMiner[1] system[25]. Specifically, to obtain the ground truth for experts, for each query, the top 30 experts from Libra, Rexa and Arnetminer are collected respectively and pooled into a single list by removing the same or ambiguous ones [31]. Then, annotators provided human judgments in terms of how many publications he/she has published, how many publications are related to the given query, how many top conference papers

[1] http://www.arnetminer.org



(a)TREC2003_TR Cosine Similarity=0.01

(b)TREC2004_TR Cosine Similarity=0.23

(c)OHSUMED_TR Cosine Similarity=0.18

Figure 2: MAP and NDCG performances for LETOR_TR.

Table 6: MAP and NDCG performances for LETOR_TR(Figure 2 in table form).

	MAP	N@1	N@3	N@5	N@10
TREC2003_TR					
RSVM	.1330	.2100	.2144	.2098	.1970
RSVMt	.0986	.1100	.1631	.1489	.1368
MTRSVM	.1100	.1850	.1730	.1703	.1656
HCDRank	**.1404**	**.2300**	**.2325**	**.2214**	**.2089**
TREC2004_TR					
RSVM	.2526	.3000	.3019	.3153	.3251
RSVMt	**.2866**	**.4138**	.3371	.3408	.3383
MTRSVM	.2464	.2966	.2949	.3038	.3151
HCDRank	.2795	.3552	**.3571**	**.3508**	**.3586**
OHSUMED_TR					
RSVM	.3541	.3483	.3255	.3141	.3044
RSVMt	.3171	.2600	.2465	.2446	.2402
MTRSVM	.3411	.3208	.3015	.2989	.2868
HCDRank	**.3758**	**.3700**	**.3654**	**.3573**	**.3459**

he/she has published, what distinguished awards he/she has been awarded. There are four rank levels (3, 2, 1, and 0), which respectively represent definite relevance $\succ$ relevance $\succ$ marginal relevance $\succ$ not relevance. To obtain the ground truth for conferences, the top 30 conferences from Libra and ArnetMiner are collected and three online resources[2] are mainly referenced for conference ranking.

In this experiment, we aim to answer the question: how can heterogeneous data be bridged for better ranking? We use the labeled data of one type of object (e.g., conferences) as the source domain and another type of object (e.g., authors) as the target domain. Thus, our goal is to transfer the conference ranking information for ranking authors.

Feature definition. We use titles of all papers published in a conference to form a conference "document", and use titles of all papers written by an author as the author's "document". Thus we can define features for each object as listed in Table 7. For each "document", there are 10 low-level content features (e.g. L1 is term frequency(tf), L5 is inverse doc frequency(idf)) and 3 high-level content features (e.g. H1 and H2 are the original and log values of BM25 score, H3 is the value of language model for IR). S1-S3 are special features for a conference which measure the number of years held and the total number of citations. S4-S7 are special features for an expert, for example, the year when his first paper has been published and the citation numbers of his all papers. Finally, we define 16 features (L1-L10, H1-H3 and S1-S3) for conference and 17 features for expert (L1-L10, H1-H3 and S4-S7).

[2]http://www.cs.ualberta.ca/~zaiane/htmldocs/ConfRanking.html and http://www3.ntu.edu.sg/home/ASSourav/crank.htm and http://www.cs-conference-ranking.org/conferencerankings/alltopics.html

Table 7: Feature definitions for expertise search.

Features	Description
L1-L10	Low-level content features, refer to [23]
H1-H3	High-level content features, refer to [23]
S1	The number of years the conference has been held
S2	The total citation of one conference during recent 5 years
S3	The total citation of one conference during recent 10 years
S4	The number of years passed since his first paper
S5	The total citation of one expert
S6	The number of papers cited more than 5 times
S7	The number of papers cited more than 10 times

We normalize the original feature vectors by query. Suppose there are $N^{(i)}$ documents $\{d_j^{(i)}\}_{j=1}^{N^{(i)}}$ with respect to i-th query, then for a feature $x_j^{(i)}$ of document $d_j^{(i)}$, after normalization, it will become

$$\frac{x_j^{(i)} - \min_k \{x_k^{(i)}\}}{\max_k \{x_k^{(i)}\} - \min_k \{x_k^{(i)}\}}, \quad k = 1, \cdots, N^{(i)}$$

Results and analysis. In this experiment, we use all the labeled conference data as the source domain, and the expert data as the target domain. In the target domain, we use one query with its corresponding documents as the labeled data and the rest as the unlabeled test data. The results reported below are averaged over all the queries. The parameter C is empirically set to 1.

As for the baseline methods, besides RSVM,RSVMt and MTRSVM, we also compare the performance of our approach with the results of two online academic search systems: Libra.msra.cn and Rexa.info, which are mainly based on unsupervised learning algorithm, e.g., the language model [30]. Table 8 shows the results of different approaches, the main observations are as follows:

1. **Ranking accuracy.** Among all the approaches, our approach HCDRank outperforms the five baselines.

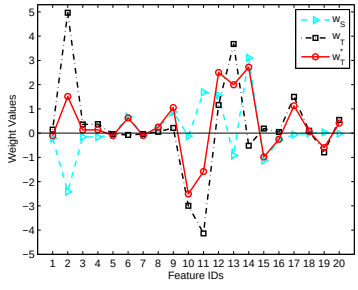
Table 8: **Performances of different approaches for expert finding.**

Approach	MAP	N@1	N@3	N@5	N@10
Libra	.5823	.3393	.2942	.3054	.3799
Rexa	.6218	.2560	.2705	.2759	.3602
RSVM	.8084	.6071	.5839	.5854	.6385
RSVMt	.8096	.5944	.6026	.5956	.6387
MTRSVM	.8059	.5791	.5796	.5810	.6379
HCDRank	**.8195**	**.6250**	**.6257**	**.6152**	**.6615**

The performances of RSVM and MTRSVM are comparable. We can also see that all learning-to-rank methods outperform the two systems. This suggests that in a specific domain, some supervised information would be very useful for improving the ranking performance.

2. **Feature analysis.** Figure 3 shows the final weight vectors learnt in this data set. We can see that the final w_T^* can exploit the data information from two domains and adjust the weights learnt from single domain data to better predict preferences in the target domain. This is the major reason why the proposed method performs best. The right table in the figure lists the top 10 features vital for knowledge transfer in this academic data set by the descending order of the absolute weight values. There are L2,L6,L9,L10 in low-level content features and H1-H3 in high-level content features and S1,S2,S4 in self-defined features.

3. **Reason for performance.** The key reason is that even in the heterogeneous network, there might be latent dependencies between the objects, some common features can still be extracted from the latent dependencies. For example, in the expertise search, authors and conferences are connected by the papers they have published. The discovered latent dependencies can be used to transfer supervised knowledge between the heterogeneous objects. Our approach can effectively discover the common latent space in the heterogeneous network, thus can achieve better performance for expertise search.



IDs	Features	Weights
14	S1	2.72
10	L10	-2.51
12	H2	2.50
13	H3	2.00
11	H1	-1.58
2	L2	1.51
17	S4	1.13
9	L9	1.05
15	S2	-0.99
6	L6	0.63

Figure 3: Feature correlation analysis in the source and the target domains. The red colored weights $\mathbf{w}_T^*$ are learnt by HCDRank; the blue and black ones ($\mathbf{w}_S$ and $\mathbf{w}_T$) are learnt from the two domains separately. The table lists top 10 features learnt from the academic data set for HCD ranking.

4.4 Results on Heterogeneous Tasks

Data set. The third experiment is for heterogeneous tasks, where we have two different ranking tasks: expert finding and best supervisor finding. The goal of expert finding is to find experts on a given topic (query), while best supervisor finding is about finding who are the best supervisors in a specific domain, which is useful for junior students to find "good" supervisors in their interested fields. An expert can be a good supervisor, but not necessarily, thus the two tasks are related but different. The goal of this experiment is to evaluate whether the proposed approach can transfer knowledge to improve a different ranking task (best supervisor finding) using training data of an existing related heterogeneous ranking task (expert finding). The demo for best supervisor finding is now online available[3].

The evaluation data set for best supervisor finding is created by collecting the feedbacks from many researchers in related domains. The data set for best supervisor finding consists of 9 most frequent queries, and for each query, we choose the top ranked 50 researchers by ArnetMiner.org and another 50 researchers who start publishing papers only in recent years (>2003, 91.6% of them are currently graduates or postdoctoral researchers). We send to each of the researchers an email, in which we list the top 50 researchers for each query, and ask for feedback on whether each candidate is the best supervisor ("yes") or not ("no"), or "not sure". Participants can also add other best supervisors. Based on the feedbacks from the participants, we organized a list for evaluating best supervisor finding. We rated each candidate person by simply counting the number of "yes"(+1) and "no" (-1) from the received feedback, and averaged the rates over the number of the corresponding definite feedbacks ("yes" and "no"). In this way, we created a relatively commonly accepted best supervisor list for each query.

Feature definition. We define 21 common features for expert finding and best supervisor finding (as shown in Table 9). Features L1-L10 and H1-H3 are scores calculated using language models, while features B1-B8 represent the expertise scores of an author from different aspects. B5-B7 are the same as S5-S7 in Table 7. In addition, we define another 32 special features for best supervisor finding. SumCo1-SumCo8 represent the overall expertise of his/her coauthors, and we average SumCo1-SumCo8 scores over the total number of his/her coauthors, denoted by AvgCo1-AvgCo8. Similarly, we consider the summation and average of the expertise of only his/her advisees through features SumStu1-SumStu8 and AvgStu1-AvgStu8. For SumStu1-SumStu8 and AvgStu1-AvgStu8, we need identify the adviser-advisee relationship between researchers. Refer to [28] for details.

Results and analysis. In this experiment, for the source domain data, we use all the labeled data from the expert finding task, and for the target domain data, we uses two sampled queries with their corresponding documents from the best supervisor finding task as the labeled data, and the rest as the unlabeled test data. Table 10 shows the performance of best supervisor finding. We see that the proposed method performs better than the baseline methods of using RSVM, RSVMt, MTRSVM and the language model based method [30]. Also we can see that all supervised learning-to-rank methods can achieve higher ranking accuracy than the unsupervised ranking method (language model).

[3] `http://bole.arnetminer.org`

Table 9: Features for expert finding and best supervisor finding.

Feature	Description
L1-L10	Low-level language model features, refer to [23]
H1-H3	High-level language model features, refer to [23]
B1	The year he/she published his/her first paper
B2	The number of papers of an expert
B3	The number of papers in recent 2 years
B4	The number of papers in recent 5 years
B5	The number of citations of all his/her papers
B6	The number of papers cited more than 5 times
B7	The numebr of papers cited more than 10 times
B8	PageRank score in academic network
SumCo1-8	The sum of coauthors' B1-B8 scores
AvgCo1-8	The average of coauthors' B1-B8 scores
SumStu1-8	The sum of his/her advisees' B1-B8 scores
AvgStu1-8	The average of his/her advisees' B1-B8 scores

Table 10: Results of best supervisor finding.

Approach	P@5	P@10	P@15	MAP	N@5	N@10
RSVM	.7714	**.8429**	.8285	.7756	.5545	.5947
RSVMt	.8000	.8286	.8476	.7837	.5923	.5999
MTRSVM	.8000	.8286	.8476	.7875	.6140	.6075
Language model	.6250	.6875	.6500	.6726	.3343	.3809
HCDRank	**.8285**	.7857	**.8571**	**.7971**	**.6189**	**.6112**

Table 11 show the top 5 best supervisors/experts for two example queries. From that, we can see the traditional expert finding algorithm is not appropriate for best supervisor finding task.

Table 11: Example lists of expert finding verse best supervisor finding.

Best Supervisor Finding	
Machine Learning	Support Vector Machine
Geoffrey E. Hinton	Bernhard Scholkopf
Sanjay Jain	Vladimir Vapnik
Michael I. Jordan	John Shawe-Taylor
Tom M. Mitchell	Alex J. Smola
Avrim Blum	Thomas Hofmann
Expert Finding	
Machine Learning	Support Vector Machine
Pat Langley	Bernhard Scholkopf
Ivan Bratko	Vladimir Vapnik
Thomas G. Dietterich	Olvi L. Mangasarian
Carl H. Smith	Chih-Jen Lin
Jaime G. Carbonell	Thorsten Joachims

5. RELATED WORK

5.1 Learning to Rank

Considerable work has been conducted for supervised learning to rank. The proposed approaches can be divided into three categories: pointwise approach, pairwise approach and listwise approach. In pointwise approaches, the ranking problem is aimed at predicting the rank level of an object. In pairwise approaches, the ranking problem can be reduced to a classification problem by comparing the rank levels of each instance pairs. Ranking SVM [15], RankBoost and RankNet [9] are three state-of-the-art algorithms in this category. In listwise approaches, the ranking problem is formulated to directly optimize some listwise performance measures of information retrieval [27, 29].

Regarding the unavailability of a large amount of training data, there is also some work on ranking by semi-supervised learning and transductive learning. For example, Duh and Kirchhoff propose a framework for ranking problem in the transductive setting. They try to extract query-specific features in order to learn a query-specific ranking function [13]. Amini et al. propose a semi-supervised rankboost algorithm [1]. Hoi and Jin propose a semi-supervised ensemble ranking with a SVM-like formulation [16]. Chen and Lu et al. propose a tree based ranking adaptation algorithm, aiming to make use of the training data from an existing domain. Specifically, they first learn a regression tree in one domain and then adapted its structure to a new domain with only a few training data [10]. Up to our knowledge, this is the most similar work to ours. However, our problem setting and the proposed approach are different from theirs. We address the ranking adaptation problem in the heterogeneous data and our approach has a clear regularized formulation.

5.2 Transfer Learning

Another related work is transfer learning, which aims to transfer knowledge from a source domain to a related target domain. Two fundamental issues in transfer learning are "what to transfer" and "when to transfer". Many approaches have been proposed by reweighting instances in source domain for the use in target domain [12]. Gao et al. propose a locally weighted ensemble framework which can utilize different models for transferring labeled information from multiple training domains [14]. Also many works have been done based on new feature representation [18, 21]. For example, Argyriou et al. propose a method to learn a shared low-dimensional representation for multiple related tasks and the task functions simultaneously [2]. Raina et al. propose to use a large amount of unlabeled data in source domain to improve the performance on target domain in which there are only few labeled data. They don't assume the two domains share the class labels or distributions [24]. Blitzer et al. proposed a structural correspondence learning approach to induce correspondences among features from source and target domains [6]. There are also other approaches which transfer information by shared parameters [7] or relational knowledge. Transfer learning techniques are widely used in classification, regression, clustering and dimensionality reduction problems.

6. CONCLUSION AND FUTURE WORK

We formally define the problem of heterogeneous cross domain (HCD) ranking and address three challenges:(1) how to formalize the problem in a *unified* and *principled* framework even when objects' types across domains are different; (2) how to transfer the knowledge of heterogeneous objects across domains; (3) how to preserve the preference relationships between instances across heterogeneous data sources. To address these, we propose a general regularized framework to discover a latent space for two domains and minimize two weighted ranking functions simultaneously in the latent space. We solve this problem by optimizing the convex upper bound of the non-continuous loss function and derive its generalization bound. Experimental results on three different genres of data sets show that the proposed approach performs better ($+1.2\% \sim +6.1\%$ in terms of MAP) than the comparison baseline methods.

There are several directions for future work. It would be interesting to develop new algorithms under the framework and to reduce the computing complexity for online application. Another issue is to extend the HCDRank framework

to combine structural information for ranking. On the Web, there are many structural information such as hyperlinks and social relationships. How to incorporate such information into the HCDRank framework is an interesting problem. Another potential issue is to apply the proposed approach to other applications (e.g., recommendation, rating, and link prediction) to further validate its effectiveness.

7. ACKNOWLEDGMENTS

Bo Wang and Songcan Chen are supported by NSFC (60773061) and NSF of Jiangsu (BK2008381). Jie Tang is supported by NSFC(60703059), National High-tech R&D Program (No. 2009AA01Z138) and Chinese Young Faculty Research Fund (No. 20070003093).

8. REFERENCES

[1] M.-R. Amini, T.-V. Truong, and C. Goutte. A boosting algorithm for learning bipartite ranking functions with partially labeled data. In *SIGIR'08*, pages 99–106, July 2008.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS'06*, pages 41–48, 2006.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[4] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *ICML'07*, pages 81–88, 2007.

[5] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *NIPS'07*, pages 129–136, 2007.

[6] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP'06*, pages 120–128, 2006.

[7] E. Bonilla, K. M. Chai, and ChrisWilliams. Multi-task gaussian process prediction. In *NIPS'08*, pages 153–160, 2008.

[8] U. Brefeld and T. Scheffer. Auc maximizing support vector learning. In *Proceedings of ICML'05 workshop on ROC Analysis in Machine Learning*, 2005.

[9] 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, 2005.

[10] K. Chen, R. Lu, C. K. Wong, G. Sun, L. Heck, and B. Tseng. Trada: tree based ranking function adaptation. In *CIKM'08*, pages 1143–1152, 2008.

[11] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *NIPS'06*, pages 281–288, 2006.

[12] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *ICML'07*, pages 193–200, 2007.

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

[14] J. Gao, W. Fan, J. Jian, and J. Han. Knowledge transfer via multiple model local structure mapping. In *KDD'08*, pages 283–291, 2008.

[15] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[16] S. C. Hoi and R. Jin. Semi-supervised ensemble ranking. In *AAAI'08*, July 2008.

[17] K. Jarvelin and J. Kekalainen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR'00*, pages 41–48, 2000.

[18] T. Jebara. Multi-task feature and kernel selection for svms. In *ICML'04*, July 2004.

[19] T. Joachims. Learning to classify text using support vector machines. *Dissertation*, 2002.

[20] T. Joachims. Training linear svms in linear time. In *KDD'06*, pages 217–226, 2006.

[21] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML'07*, pages 489–496, July 2007.

[22] J. Liu, S. Ji, and J. Ye. Accelerated multi-task feature learning. In *UAI'09*, June 2009.

[23] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR 2007, in conjunction with SIGIR 2007*, 2007.

[24] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML'07*, pages 759–766, June 2007.

[25] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.

[26] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. *Singular value decomposition and principal component analysis*, pages 91–109. Kluwer: Norwell, MA, 2003.

[27] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR'07*, pages 391–398, 2007.

[28] Z. Yang, J. Tang, B. Wang, J. Guo, J. Li, and S. Chen. Expert2bole: From expert finding to bole search. In *KDD'09*, 2009.

[29] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR'07*, pages 271–278, 2007.

[30] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM'01*, pages 403–410, 2001.

[31] J. Zhang, J. Tang, and J. Li. Expert finding in a social network. In *DASFAA'07*, pages 1066–1069, 2007.

9. APPENDIX: DERIVATION OF THE EQUIVALENT CONVEX FORMULATION

We give a brief proof on the equivalence between Eq. 7 and Eq. 8. We follow the same structure as the proof of equation equivalence in [2]. For easy explanation, we denote the objective functions in Eq. 7 and Eq. 8 as $\mathcal{E}(W,U)$ and $\mathcal{R}(M,D)$ respectively.

THEOREM 2. *Problem of* $\min\{\mathcal{E}(W,U) : U^\top U = I\}$ *is equivalent to the problem* $\min\{\mathcal{R}(M,D) : D \succeq 0, trace(D) \leq 1, range(M) \subseteq range(D)\}$.

PROOF. The correspondence between the two problems is $M = UW$ and $D = U\text{Diag}(\frac{||a^i||_2}{||W||_{2,1}})U^\top$. Let a^i be the i-th row of W, then $\|a^i\|_2 = \|M^\top u_i\|_2$. So

$$\begin{aligned}
&\sum_{t=1}^{2} \langle \alpha_t, D^+ \alpha_t \rangle = \text{trace}(M^\top D^+ M) \\
&= \|W\|_{2,1} \ \text{trace}(M^\top U \text{Diag}(\|M^\top u_i\|_2)^+ U^\top M) \\
&= \|W\|_{2,1} \ \text{trace}(\sum_{i=1}^{d} (\|M^\top u_i\|_2)^+ M^\top u_i u_i^\top M) \\
&= \|W\|_{2,1} \sum_{i=1}^{d} \|M^\top u_i\|_2 \\
&= \|W\|_{2,1}^2
\end{aligned}$$

Therefore, $\min_{M,D} \ \mathcal{R}(M,D) \leq \min_{W,U} \ \mathcal{E}(W,U)$.
On the other side, let $D = U\text{Diag}(\lambda_i)U^\top$, then

$$\begin{aligned}
&\sum_{t=1}^{2} \langle \alpha_t, D^+ \alpha_t \rangle = \text{trace}(M^\top U \text{Diag}(\lambda_i^+) U^\top M) \\
&= \text{trace}(\text{Diag}(\lambda_i^+) W W^\top) \geq \|W\|_{2,1}^2
\end{aligned}$$

Hence, $\min_{M,D} \ \mathcal{R}(M,D) \geq \min_{W,U} \ \mathcal{E}(W,U)$. So the two problems are equivalent. □