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 heterogeneous 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM ’09, November 2–6, 2009, Hong Kong, China.
Copyright 2009 ACM 978-1-60558-512-3/09/11 ...$5.00.

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 resources2). 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
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 a 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% ~ +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.

2. PROBLEM FORMULATION

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

**Input:** Let \( \mathcal{X}_S \subset \mathbb{R}^d \) be the instance space of the source domain in which \( d \) is the number of features and \( \mathcal{Y}_S = \{r_{s1}, r_{s2}, \ldots, r_{sp}\} \) denotes a set of rank levels where \( p \) is the number of rank levels in the source domain. The rank levels have: \( r_{s1} \succ r_{s2} \succ \cdots \succ r_{sp} \), where \( \succ \) denotes the preference relationship. The labeled data in the source domain is denoted by \( \mathcal{L}_S = \{(q^k_S, x^k_S, y^k_S)\}_{k=1}^n \). That is, for query \( q^k_S \), \( x^k_S \in \mathcal{X}_S \) and \( y^k_S \) are the corresponding labels where \( x^k_S \in \mathcal{X}_S \), \( y^k_S \in \mathcal{Y}_S \) and \( \mathcal{L}_S \) is the total number of instances related to this query. Further, for the target domain, let \( \mathcal{X}_T \subset \mathbb{R}^d \) be the instance space and \( \mathcal{Y}_T = \{r_{t1}, r_{t2}, \ldots, r_{tp}\} \) is the set of rank levels. There are two parts of data in the target domain: \( \mathcal{S} = \{(q^k_T, x^k_T)\}_{k=1}^m \) represents the unlabeled test data in which \( x^k_T \in \mathcal{X}_T \) and \( \mathcal{L}_T = \{(q^k_T, x^k_T, y^k_T)\}_{k=1}^n \) represents the labeled data where \( x^k_T \in \mathcal{X}_T \) and \( y^k_T \in \mathcal{Y}_T \). 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-
3.2 The General Framework: HCDRank

Given the labeled training data from the target domain \( \mathcal{L}_T = \{ (q^T_i, \tilde{x}^T_i, y^T_i) \}_{i=1}^{n_T} \), we aim to learn a ranking function \( f_T \) which can correctly predict the preference relationships between instances for each query \( q^T_i \), i.e. \( f_T(x^T_i) > f_T(x^T_j) \) : \( \forall y^T_i > y^T_j \). 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:

\[
\min_{f_T} \mathcal{O}(f_T, \mathcal{L}_T) = R(f_T, \mathcal{L}_T) + \eta \mathcal{E}(f_T)
\]

\[
= \sum_{k=1}^{n_T} \sum_{y^T_k < y^T_j} \mathbb{I}\{f_T(x^T_k) > f_T(x^T_j)\} + \eta \mathcal{E}(f_T)
\]

where \( \mathbb{I}[\pi] \) is the indicator function returning 1 when \( \pi \) is true and 0 otherwise; \( R(f_T, \mathcal{L}_T) \) counts the number of mis-ranked pairs in the target domain; \( \eta \) 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 \( \phi : \mathbb{R}^d \rightarrow \mathbb{R}^d \) 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:

\[
\min_{f_S, f_T, \phi} R_\phi(f_S, \mathcal{L}_S) + CR_\phi(f_T, \mathcal{L}_T) + \lambda J_\phi(f_S, f_T)
\]

\[
= \sum_{k=1}^{n_S} \sum_{y^S_k < y^S_j} \mathbb{I}\{f_S(\phi(x^S_k)) > f_S(\phi(x^S_j))\}
\]

\[
+ C \sum_{k=1}^{n_T} \sum_{y^T_k < y^T_j} \mathbb{I}\{f_T(\phi(x^T_k)) > f_T(\phi(x^T_j))\}
\]

\[
+ \lambda J_\phi(f_S, f_T)
\]

where \( J_\phi(f_S, f_T) \) is a penalty for the complexity of the HCD ranking model, \( \lambda \) 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 \( \phi \) 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 \( \phi \). 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 \( \langle \cdot, \cdot \rangle \) indicates the inner product. By substituting it into Eq. 1 we have

\[
\mathcal{O}(f_T, \mathcal{L}_T) = \sum_{k=1}^{n_T} \sum_{y^T_k < y^T_j} \mathbb{I}\{(w_T, x^T_k - x^T_j) > 0\} + \eta \mathcal{E}(f_T)
\]

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 \( q^T_i (k = 1, \cdots, n_T) \), given an instance pair \( x^T_k, x^T_j \) from different rank levels and their corresponding labels \( y^T_k, y^T_j \), we create a new instance

\[
\begin{cases}
+1 & \text{if } y^T_k > y^T_j \\
-1 & \text{if } y^T_k < y^T_j
\end{cases}
\]

Then we can get a new training data set consisting of instance pairs in the target domain \( \mathcal{L}'_T = \{ (x^T_k - x^T_j, z^T_k, z^T_j) \}_{j=1}^{n_T} \). For the source domain, we can make the same assumption and use the parallel notations \( w_S \) and \( \mathcal{L}'_S = \{ (x^S_k - x^S_j, z^S_k, z^S_j) \}_{i=1}^{n_S} \). Finally, we can rewrite the objective function by optimizing the convex upper bound of the original loss as:

\[
\min_{w_S, w_T, \phi} \sum_{i=1}^{n_S} \left[ 1 - z^S_i (w_S, \phi(x^S_i) - \phi(x^S_j)) \right] + \\
+ C \sum_{i=1}^{n_T} \left[ 1 - z^T_i (w_T, \phi(x^T_i) - \phi(x^T_j)) \right] + \\
+ \lambda J_\phi(w_S, w_T)
\]

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 \( x^T_i U U^T x_j \) using the matrix. Such parameterization is equivalent to projecting every example \( x \) onto a latent space spanned by \( \phi : x \rightarrow U^T x \). With the transformation function, we can redefine the loss function, for example, by replacing the first term in Eq. 5 with:

\[
\sum_{i=1}^{n_S} \left[ 1 - z^S_i (w_S, U^T (x^S_k - x^S_j)) \right] + \\
+ \lambda J_\phi(w_S, w_T)
\]

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 $W = [w_2, w_T]$ is a $d \times 2$ matrix with the first column corresponding to $w_2$ 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:

$$
\begin{align*}
\min_{w_2,w_T,U} & \sum_{i=1}^{n_1} [1 - z_{S_i}(w_2, U^T (x_{S_i}^0 - x_{S_i}^b))] + \\
& + C \sum_{i=1}^{n_2} [1 - z_{T_i}(w_T, U^T (x_{T_i}^0 - x_{T_i}^b))] + \lambda \|W\|_{2,1} \quad (7)
\end{align*}
$$

subject to $U^TU = I$ where $U$ denotes an orthogonal constraint which makes the projection matrix $U$ unique.

**Learning algorithm.** Directly solving the objective function (including solving parameters $w_2$, $w_T$, $U$ 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.)

$$
\begin{align*}
\min_{M,D} & \sum_{i=1}^{n_1} [1 - z_{S_i}(\alpha_1, x_{S_i}^0 - x_{S_i}^b)] + \\
& + C \sum_{i=1}^{n_2} [1 - z_{T_i}(\alpha_2, x_{T_i}^0 - x_{T_i}^b)] + \lambda \|M\|_{2,1} \quad (8)
\end{align*}
$$

subject to $D \geq 0$, $\text{trace}(D) \leq 1$, $\text{range}(M) \subseteq \text{range}(D)$

where $M = [\alpha_1, \alpha_2]$, $U = UW$, $D = U\text{Diag}(\|a_i\|_2)U^T$ 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 range$(X) = \{x | Xx = x, \text{for some } x \in \mathbb{R}^k\}$. The trace constraint of $D$ is imposed because if $D$ is set to $\infty$, 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 $\alpha_1$ and $\alpha_2$ (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. $D = U\Sigma V^T$; 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 labled 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.

### 3.4 Generalization Bound

First, let a domain be defined by two terms: the distribution $D$ on instance space $X$, and a ranking function $f : X \rightarrow \{r_1, r_2, \ldots, r_p\}$. Then source and target domains are denoted by $(\mathcal{DS}, f_S)$ and $(\mathcal{DT}, f_T)$ respectively. Let $\ell_S(h)$ and $\ell_T(h)$ denote the source and target risks. Correspondingly, $\ell_S(h)$ and $\ell_T(h)$ are the empirical risks.

An equivalent formulation for Eq. 7 is as follows:

$$
\begin{align*}
\min_{w_2,w_T, U} & \sum_{i=1}^{n_1} [1 - z_{S_i}(w_2, U^T (x_{S_i}^0 - x_{S_i}^b))] + \\
& + C \sum_{i=1}^{n_2} [1 - z_{T_i}(w_T, U^T (x_{T_i}^0 - x_{T_i}^b))] + \lambda \|W\|_{2,1} \quad (9)
\end{align*}
$$

subject to $\|W\|_{2,1} \leq z$, $U^TU = I$

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

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

$$
\ell_{\theta}(h) = \theta\ell_T(h) + (1 - \theta)\ell_S(h)
$$

where $\theta = \frac{1}{1-\theta}$. $\ell_{\theta}(h)$ and $\ell_S(h)$ are the empirical and true weighted risk respectively. Hereafter, we will analyze the objective function in the formulation of Eq. 10.

---

**Algorithm 1: HCDRank for transfer ranking**

**Input:** Training set: $\mathcal{L}_s \cup \mathcal{L}_T$; Test set: $\mathcal{S}$

**Output:** Ranking function $f^*_T = (w^*_T, x)$ and the predicted preferences over test data: $(y_i^*)_{i=1}^n$

**Initialization:** $D = \frac{I + d \cdot d}{d}$

**Step 1: Latent Space Finding**

1: while not reached maximal iteration number $T$ do
2: \hspace{1em} $\alpha_1 = \arg\min_{\alpha_1, \alpha_2} \left\{ \sum_{i=1}^{n_1} [1 - z_{S_i}(\alpha_1, x_{S_i} - x_{S_i}^b)] + \lambda(\alpha, D^+\alpha) \right\}$
3: \hspace{1em} $\alpha_2 = \arg\min_{\alpha_1, \alpha_2} \left\{ \sum_{i=1}^{n_2} [1 - z_{T_i}(\alpha_2, x_{T_i} - x_{T_i}^b)] + \lambda(\alpha, D^+\alpha) \right\}$
4: \hspace{1em} $M = [\alpha_1, \alpha_2]$
5: \hspace{1em} set $D = \frac{(DM^+M^+)^{-\frac{1}{2}}}{\text{trace}(M^+M^+)}$
6: end while
7: Apply SVD decomposition on $D$, $D = U\Sigma V^T$
8: Construct $U$ by the eigenvectors corresponding to the first and second biggest eigenvalues of $D$

**Step 2: Learning in Latent Space**

9: $w_T^* = \arg\min_{w_T} \left\{ \sum_{i=1}^{n_2} [1 - z_{T_i}(w_T, U^T (x_{T_i}^0 - x_{T_i}^b))] + \\
\hspace{1.5em} + C \sum_{i=1}^{n_2} [1 - z_{T_i}(w_T, U^T (x_{T_i}^0 - x_{T_i}^b))] + \lambda \|w\|^2 \right\}$
10: for $i = 1$ to $n$ do
11: \hspace{1.5em} $y_i = (w_T^*, U^T i)$
12: end for
THEOREM 1. Let \( H \) be a hypothesis space of VC-dimension \( d \). Let \( D_S \) and \( D_T \) be unlabeled samples of size \( m' \) each, drawn from \( D_S \) and \( D_T \) respectively, and \( d_{H,S} \) is the empirical distance between them. Let \( L = L_S \cup L_T \) be the labeled samples of size \( m \) generated by drawing \((1 - \beta)m\) points from \( D_S \) and \( \beta m \) points from \( D_T \), labeling them according to \( fs \) and \( f_T \) respectively. For each ranking function \( h \) with zero training risk, if \( \hat{h} \in H \) is the empirical minimizer of \( \epsilon_S(h) \) on \( L \), then with probability of at least \( 1 - \delta \) (over the choice of the samples) [5, 15]

\[
e_T(h) < \frac{\gamma}{2m}\left( d \log \left( \frac{8(c(m-1))}{8} \right) \log(32(\beta m-1)) + \log \left( \frac{8(\beta m - 1)}{8} \right) \right)
+ 2\sqrt{\frac{c^2}{\beta} + \frac{(1-\beta)^2}{2}}\frac{}{}\frac{1}{\gamma} + \frac{2}{\gamma} \frac{\log(\log(m))}{m} + \frac{c\gamma}{\log(m)} + \frac{\gamma}{\log(m)}
\]

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

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 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.

\[
AP = \frac{\sum_n P@n \cdot \#(\text{document } n \text{ is relevant})}{\#(\text{relevant documents})}
\]

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:

\[
NDCG = Z_n \sum_j^{n} \frac{2^r(j) - 1}{\log(1 + j)}
\]

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 \( \mathcal{L}_S \cup \mathcal{L}_T \). The third comparison method is MTRSVM which is a multi-task feature learning approach using ranking SVM hinge loss [2].

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 SVMlight [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 HTS) 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, TREC2003, TREC2004, and OHSUMED, respectively. 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 only contains feature categories A and B with queries 1-25 which correspond to features for document contents; while the target domain of TREC2003 only consists of feature categories B,
and OHSUMED document-pairs corresponding to a query.

Specifically, we have the following observations:

- It is highly possible that rank level of the two documents are also similar with moving the same or ambiguous ones [31]. Then, annotators selected from the query log of the ArnetMiner system [25].

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED), and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED), and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED), and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED), and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED), and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

For ease of implementation, in the experiments, we still define each instance with a vector of 44 dimensions (TREC) or 25 dimensions (OHSUMED), and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

Available features were 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 $L_S$, and randomly sample 20% of the queries and the related documents from the target domain as the training data $L_T$ 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 $S$.

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 $q_1d_1$ from the source domain is similar to that of query-document pair $q_2d_2$ 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.

### 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 system [25]. Specifically, to obtain the ground truth for experts, for each query, the top 30 experts from Libra, Reza 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

---

### Table 3: Description of features in LETOR data set.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Feature Categories</th>
<th>Description</th>
<th>#Features</th>
<th>Feature IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC</td>
<td>A</td>
<td>low-level content features</td>
<td>15</td>
<td>(25-35)</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>part of high-level content features</td>
<td>6</td>
<td>(15,16,19,20,23,24)</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>part of high-level content features</td>
<td>7</td>
<td>(1,17,18,21,22,25,26)</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>hyperlink features</td>
<td>7</td>
<td>(6-8,14,36-38)</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>hybrid features</td>
<td>8</td>
<td>(13,27,39-44)</td>
</tr>
<tr>
<td>OHSUMED</td>
<td>F</td>
<td>low-level content features for title</td>
<td>10</td>
<td>(1-10)</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>low-level content features for abstract</td>
<td>10</td>
<td>(1-20)</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>high-level content features</td>
<td>5</td>
<td>(21-25)</td>
</tr>
</tbody>
</table>

### Table 4: Data characteristics of LETOR_TR data set.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Query IDs</th>
<th>Features</th>
<th>#Doc</th>
<th>#D/Q</th>
<th>#Dp/Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC2003_TR</td>
<td>25-{1-25}</td>
<td>AB</td>
<td>24079</td>
<td>963</td>
<td>6450</td>
</tr>
<tr>
<td></td>
<td>25-{26-50}</td>
<td>BCDE</td>
<td>25092</td>
<td>1004</td>
<td>13761</td>
</tr>
<tr>
<td>TREC2004_TR</td>
<td>38-{1-107}</td>
<td>AE</td>
<td>37154</td>
<td>978</td>
<td>5969</td>
</tr>
<tr>
<td></td>
<td>37-{111-221}</td>
<td>BCDE</td>
<td>37016</td>
<td>1000</td>
<td>5690</td>
</tr>
<tr>
<td>OHSUMED_TR</td>
<td>56-{1-56}</td>
<td>FH</td>
<td>8136</td>
<td>145</td>
<td>5720</td>
</tr>
<tr>
<td></td>
<td>50-{57-106}</td>
<td>GH</td>
<td>8004</td>
<td>160</td>
<td>5239</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Data Set</th>
<th>RSVM</th>
<th>RSVMt</th>
<th>MTRSVM</th>
<th>HCDRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC2003_TR</td>
<td>744</td>
<td>973</td>
<td>6180</td>
<td>6417</td>
</tr>
<tr>
<td>TREC2004_TR</td>
<td>83</td>
<td>1125</td>
<td>6810</td>
<td>7548</td>
</tr>
<tr>
<td>OHSUMED_TR</td>
<td>647</td>
<td>820</td>
<td>7146</td>
<td>7477</td>
</tr>
</tbody>
</table>
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 inverse doc frequency(idf)) and 3 high-level content features (e.g. H1, H2 and S4–S7).

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

\[
\frac{x_j(i) - \min_k \{x_k(i)\}}{\max_k \{x_k(i)\} - \min_k \{x_k(i)\}}, \quad k = 1, \ldots, N(i)
\]

**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 Reza.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.

<table>
<thead>
<tr>
<th>Approach</th>
<th>MAP</th>
<th>N@1</th>
<th>N@3</th>
<th>N@5</th>
<th>N@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libra</td>
<td>0.5825</td>
<td>0.3939</td>
<td>0.2942</td>
<td>0.3654</td>
<td>0.3799</td>
</tr>
<tr>
<td>Reoxa</td>
<td>0.6218</td>
<td>0.2560</td>
<td>0.2705</td>
<td>0.2759</td>
<td>0.3602</td>
</tr>
<tr>
<td>RSVM</td>
<td>0.8084</td>
<td>0.6701</td>
<td>0.5839</td>
<td>0.5554</td>
<td>0.6385</td>
</tr>
<tr>
<td>RSVMt</td>
<td>0.896</td>
<td>0.5944</td>
<td>0.6026</td>
<td>0.5596</td>
<td>0.6387</td>
</tr>
<tr>
<td>MTR SVM</td>
<td>0.8059</td>
<td>0.5791</td>
<td>0.5796</td>
<td>0.5810</td>
<td>0.6379</td>
</tr>
<tr>
<td>HCDRank</td>
<td>0.8195</td>
<td>0.6250</td>
<td>0.6257</td>
<td>0.6152</td>
<td>0.6615</td>
</tr>
</tbody>
</table>

The performances of RSVM and MTR SVM 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, 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.

Figure 3: Feature correlation analysis in the source and the target domains. The red colored weights $w_S$ are learnt by HCDRank; the blue and black ones ($w_S$ and $w_T$) 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.

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. SumCol-SumCo8 represent the overall expertise of his/her coauthors, and we average SumCol-SumCo8 scores over the total number of his/her coauthors, denoted by AvgCol-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. Interested readers can 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 use 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, MTR SVM and the language model based method [30]. Also we can see that all supervised learning-to-rank methods can achieve higher ranking accuracy than

![http://bole.arnetminer.org](http://bole.arnetminer.org)
5.2 Transfer Learning

Our approach has a clear regularized formulation. 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% 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 the unsupervised ranking method (language model).

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

<table>
<thead>
<tr>
<th>Expert Finding</th>
<th>Best Supervisor Finding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>Geoffrey E. Hinton</td>
<td>Bernhard Scholkopf</td>
</tr>
<tr>
<td>Sanjay Jain</td>
<td>Vladimir Vapnik</td>
</tr>
<tr>
<td>Michael I. Jordan</td>
<td>John Shawe-Taylor</td>
</tr>
<tr>
<td>Tom M. Mitchell</td>
<td>Alex J. Smola</td>
</tr>
<tr>
<td>Avrim Blum</td>
<td>Thomas Hofmann</td>
</tr>
</tbody>
</table>

Table 10: Results of best supervisor finding.

<table>
<thead>
<tr>
<th>Approach</th>
<th>P@5</th>
<th>P@10</th>
<th>P@15</th>
<th>MAP</th>
<th>N@5</th>
<th>N@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSVM</td>
<td>.7714</td>
<td>.8429</td>
<td>.8285</td>
<td>.7756</td>
<td>.3545</td>
<td>.5947</td>
</tr>
<tr>
<td>RSVMT</td>
<td>.8000</td>
<td>.8286</td>
<td>.8476</td>
<td>.7875</td>
<td>.5923</td>
<td>.5999</td>
</tr>
<tr>
<td>MTRSVM</td>
<td>.8000</td>
<td>.8286</td>
<td>.8476</td>
<td>.7875</td>
<td>.6140</td>
<td>.6075</td>
</tr>
<tr>
<td>Language model</td>
<td>6250</td>
<td>6.875</td>
<td>6.500</td>
<td>6.726</td>
<td>3.343</td>
<td>3.809</td>
</tr>
<tr>
<td>HCDRank</td>
<td>.8285</td>
<td>.7857</td>
<td>.8571</td>
<td>.7971</td>
<td>.6189</td>
<td>.6112</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LT-L10</td>
<td>Low-level language model features, refer to [23]</td>
</tr>
<tr>
<td>H1-H3</td>
<td>High-level language model features, refer to [24]</td>
</tr>
<tr>
<td>B1</td>
<td>The year he/she published his/her first paper</td>
</tr>
<tr>
<td>B2</td>
<td>The number of papers of an expert</td>
</tr>
<tr>
<td>B3</td>
<td>The number of papers in recent 2 years</td>
</tr>
<tr>
<td>B4</td>
<td>The number of papers in recent 5 years</td>
</tr>
<tr>
<td>B5</td>
<td>The number of citations of all his/her papers</td>
</tr>
<tr>
<td>B6</td>
<td>The number of papers cited more than 5 times</td>
</tr>
<tr>
<td>B7</td>
<td>The number of papers cited more than 10 times</td>
</tr>
<tr>
<td>B8</td>
<td>PageRank score in academic network</td>
</tr>
<tr>
<td>SumCo1-8</td>
<td>The sum of coauthors’ B1-B8 scores</td>
</tr>
<tr>
<td>AvgCo1-8</td>
<td>The average of coauthors’ B1-B8 scores</td>
</tr>
<tr>
<td>SumStu1-8</td>
<td>The sum of his/her advisees’ B1-B8 scores</td>
</tr>
<tr>
<td>AvgStu1-8</td>
<td>The average of his/her advisees’ B1-B8 scores</td>
</tr>
</tbody>
</table>
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. 2007003093).

8. REFERENCES


9. APPENDIX: DERIVATION OF THE EQUIV-ALENT 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. We follow the same structure as the proof of [2].

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

$$\sum_{t=1}^2 (\alpha_t, D^+ a_t) = \text{trace}(M^\top D^+ M)$$

$$= \|W\|_{2,1} \text{trace}(M^\top \text{Diag}(\|M^\top u^t\|_2) U^\top M)$$

$$= \|W\|_{2,1} \text{trace}(d \sum_{i=1}^d (\|M^\top u^t\|_2^2 + M^\top u^t M))$$

$$= \|W\|_{2,1} \sum_{i=1}^d \|M^\top u^t\|_2$$

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

$$\sum_{t=1}^2 (\alpha_t, D^+ a_t) = \text{trace}(M^\top U \text{Diag}(\lambda^t) U^\top M)$$

$$= \text{trace}(\text{Diag}(\lambda^t) WW^\top) \geq \|W\|_{2,1}^2$$

Hence, $\min_{M,D} R(M, D) \geq \min_{W,U} E(W, U)$. Finally, we get $\min_{M,D} R(M, D) = \min_{W,U} E(W, U)$. □