Democracy is Good for Ranking: Towards Multi-view Rank Learning and Adaptation in Web Search

Wei Gao
Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
wgao@qf.org.qa

Pei Yang
Department of Computer Science
South China University of Technology
Guangzhou, China
yangpei@scut.edu.cn

ABSTRACT

Web search ranking models are learned from features originated from different views or perspectives of document relevancy, such as query dependent or independent features. This seems intuitively conformant to the principle of multi-view approach that leverages distinct complementary views to improve model learning. In this paper, we aim to obtain optimal separation of ranking features into non-overlapping subsets (i.e., views), and use such different views for rank learning and adaptation. We present a novel semi-supervised multi-view ranking model, which is then extended into an adaptive ranker for search domains where no training data exists. The core idea is to proactively strengthen view consistency (i.e., the consistency between different rankings each predicted by a distinct view-based ranker) especially when training and test data follow divergent distributions. For this purpose, we propose a unified framework based on listwise ranking scheme to mutually reinforce the view consistency of target queries and the appropriate weighting of source queries that act as prior knowledge. Based on LETOR and Yahoo Learning to Rank datasets, our method significantly outperforms some strong baselines including single-view ranking models commonly used and multi-view ranking models that do not impose view consistency on target data.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Multi-view rank learning; rank adaptation; view consistency

1. INTRODUCTION

The capability of combining a large number of features in an optimal way is highly desirable for search engines [29]. Learning to rank or rank learning for Information Retrieval (IR) aims to learn how to combine the predefined relevance features properly for better ranking search results.

Typically, ranking features consist of hundreds of relevance scores derived from various retrieval relevance models, each of which corresponds to a permutation of search results from some particular perspective. For example, query dependent features, such as frequencies of the query terms in the document and BM25, are constructed from the perspective of query-document relevancy, while query independent features, such as PageRank and HostRank, are provided by link analysis algorithms depending on the connectivity of documents only. It can be expected that these distinct views of features, although possibly redundant sometimes, if combined correctly, may afford strong complementarity to each other and result in considerable improvement for ranking.

Such natural feature partition seems intuitively conformant to the philosophy of multi-view learning [6, 1, 41]. It was found that having multiple representations instead of combining all features into one view can improve classification performance when many unlabeled examples are available in addition to the labeled ones. For instance, the idea of co-training [6] is to train one learner on each view of the labeled data, and then each learner iteratively labels the unlabeled subsets of data where it has the highest confidence in its prediction. Since the views are independent, the newly labeled examples by one learner can give the other learner novel information for improving its model parameters.

Moreover, rank learning requires large training set that is very expensive or time-consuming to obtain. Co-training follows semi-supervised fashion where it needs only partially labeled data, and unlabeled data can also participate in training. In the competitive Web search market, it would be much desirable to quickly deploy ranking models into other search domains to support different verticals or search in the market of different languages where training data are rarely available. As a result, we need to extend in-domain ranking to rank adaptation for good out-of-domain performance by bridging domain gap [23, 13, 39, 12, 21, 22]. The semi-supervised nature of co-training-based algorithms makes them naturally extendible to cross-domain setting.

Although classification methods based on co-training are common [16, 17, 7, 32], the study of multi-view rank learning is almost absent. Existing multi-view classifiers hardly could be applied for ranking directly because (1) the concentration of ranking is on the order of a list of documents rather than their absolute class labels; (2) the confidence measure of a ranker and the consistency between rankers on distinct views are radically different from that of classifiers,

WSDM'14, February 24–28, 2014, New York, New York, USA.

http://dx.doi.org/10.1145/2556195.2556267.

which may require estimating the permutation probability of output rankings; (3) there is no explicit view boundary or feature definition provided in most ranking benchmark data sets. In addition, multi-view methods assume that different view-based learners give compatible predictions on the same instance with certain probability [6], but such view consistency assumption is largely violated in a cross-domain setting where training and test data are drawn from different distributions. All these render multi-view approach for rank learning and adaptation practically challenging.

In this paper, we investigate a novel and effective solution to tackle these challenges based on typical listwise ranking scheme [9, 40, 34, 36, 42], specifically ListMLE [40]. We extend an automatic feature decomposition approach for co-training [15], by which we simultaneously obtain the optimal feature split and exploit such multiple views for rank learning and adaptation. Different from [15] which is focused on class conditionally independent assumption, we argue that enhancing view consistency across domains can improve the adaptation effectiveness for multi-view ranking algorithms. We propose a unified ranking framework that incorporates listwise loss, listwise view consistency on target queries and the appropriate weighting of source queries. To deal with the lack of view agreement in adaptation setting, our method adopts an iterative procedure to mutually reinforce the view consistency of ranking on target queries and the weighting of source queries. Intuitively, since transferrable source queries likely have better view consensus in both domains, imposing view consensus on target queries may result in a stronger model being helpful to weigh out those transferrable source queries that can be used to further improve view consistency in target domain. With LETOR3.0 and Yahoo Learning to Rank datasets, experimental results show that our method makes significant improvements over the baselines.

The rest of this paper is organized as follows: Section 2 reviews the related work; Section 3 gives the preliminaries of using multiple views for ranking and adaptation; Section 4 presents the proposed multi-view ranking method; Section 5 descries experiments and discusses results; Section 6 draws conclusion and looks ahead to future work.

2. RELATED WORK

In this section, we briefly review ranking, domain adaptation, multi-view learning and how they were related.

2.1 Rank Learning and Adaptation

Rank learning is to optimize a ranking function given data consisting of queries, the retrieved documents and their relevance judgements. Given a new query, the learned function is used to predict the order of retrieved documents. Based on input spaces, three categories of approaches have been proposed, namely pointwise, pairwise and listwise approach. Listwise scheme [9, 40, 34, 36, 42] addresses ranking by considering the entire list of documents associated with the same query as instance and considers positional information in the loss function which intuitively makes more sense with the nature of ranking [29]. Our algorithm is listwise. A comprehensive survey on learning to rank is in [29].

It is worth mentioning that for saving labeling cost of ranking data, some studies explored semi-supervised rank learning with only partially labeled data in transductive [18] or inductive [2] manner. Multi-view approach is inherently semi-supervised and can be practiced in either way.

Domain adaptation [5, 24, 25, 27, 43] aims to save in-domain labeling expense by using out-of-domain training data, and tries to bridge the dataset shift [31] between distributions of two domains by using their common information. It is difficult to directly use classifier adaptation for ranking as rank adaptation is comparatively more challenging [23]. Therefore, rank adaptation received more and more attention recently [23, 13, 39, 12, 21, 22, 8]. In [13], the Gradient Boosting Tree structure of ranker learned from source data was adjusted using some target labeled data. In [23], prior knowledge was learned from source domain and the similarity of parameters between two domains was considered when training the target ranker. Both instance-based and feature-based adaptation were conducted for ranking with the help of a few target training data in [12]. Rank adaptation was also dealt with in [39] by bridging heterogeneous domains via some shared latent space. In [21], rankers were learned from two domains separately and then interpolated for a stronger model. Various instance-weighting schemes were proposed in [22] for adaptation with a more relaxed setting where no target ranking data is labeled. More recently, active selection of target queries was studied by [8] for improving rank adaptation performance. All these rank adaptation methods are of single view.

2.2 Multi-view Learning

Multi-view learning is a semi-supervised framework that utilizes the consensus among learners trained on independent views of the same problem to improve the model's overall performance. It holds two basic assumptions, i.e., conditional independence assumption and view consistency assumption. Early approaches include co-training [6] and its variants [41, 16]. Then the idea was widely explored or extended later on [1, 3, 17, 7, 32]. For example, the unnecessarily strong view-independence assumption was relaxed in [1], which then suggested that the disagreement rate of two independent hypotheses upper bounds the error rate of either hypothesis. It was also proved that a weaker expanding property called ϵ-expandability on the multi-view data distribution is sufficient for co-training to work [3].

Co-training also assumes that all labels on the examples with non-zero probability are consistent with both hypotheses of each view [6]. Under this view consistency assumption, some algorithms impose consensus constraint between the predictors, each trained from one view of data, to improve the learning performance [35, 19, 20, 28]. A co-regularization approach [35] was proposed to learn a multi-view classifier from partially labeled data using a view-consensus-based regularization term. A supervised algorithm named SVM-2K [19] was proposed which imposed a similarity constraint between two distinct view-based SVMs. A two-view transductive SVM [28] was constructed by extending the work of [19] for utilizing the large set of unlabeled data available. A two-view learning algorithm [20] was studied by using stochastic agreement regularization based on a constrained EM. An automatic feature decomposition algorithm [15] was presented for pseudo multi-view learning where the original data has only one view.

All these methods deal with classification problem and treat distinct domains indiscriminately. We are focused on multi-view rank learning problem and our setting is more general where training and test data can come from related but different domains.

2.3 Hybrid Approach

The combination of multi-view learning and adaptation is potentially powerful but not well studied. Ganchev et al. [20] applied the proposed two-view semi-supervised learner to cross-domain sentiment classification, but they did not consider bridging domain gap explicitly in the model. Chen et al. [14] presented a variant of co-training for domain adaptation based on two logistic regression classifiers, and feature correlation between two domains was employed for feature selection that aimed to bridge the gap, but view consistency factor was not considered. Zhang et al. [44] integrated multi-view learning and transfer learning with a large margin approach, where source training data were used to learn a large margin classifier and the data from both domains were used to impose consistencies among multiple views. However, they did not co-relate domain gap to view consistency in the model which is important to adaptation. All these algorithms are based on classification models.

Multi-view semi-supervised learning was explored for ranking multilingual documents [37] where each view corresponds to one language. But it was for bipartite ranking and its feature space adopted bag-of-words representation, which is equivalent to classification. Adaptation was not considered.

Our method combines multi-view learning, ranking and adaptation in a unified framework to boost cross-domain ranking performance. There are significant distinctions from existing work: (1) we work on multi-view ranking which differs radically from classification; (2) we attempt to co-relate the factors of domain commonality and view consistency, and focus on mutually reinforcing the two components to improve rank transfer. To our knowledge, there is no previous work based on multi-view approach for rank adaptation.

3. PRELIMINARIES

Let rank training set contain n queries $Q = \{q_i\}_{i=1}^{n}$, and each q_i is associated with a list of documents represented by feature vectors $\mathbf{x}_i = \{\vec{x}_{ij}\}_{j=1}^{m_i}$ and a list of rank labels $\mathbf{y}_i = \{y_{ij}\}_{j=1}^{m_i}$ (each corresponds to a document), where m_i is the number of documents of q_i, $\vec{x}_{ij} \in R^N$ is a feature vector consisting of N pre-defined rank features of document ij, and y_{ij} is its relevance judgement (e.g., highly relevant, relevant or irrelevant). The goal is to optimize a ranking function $f : \mathbf{x}_i \rightarrow \pi_i$ that maps the input document list $\mathbf{x}_i$ into a permutation π_i of documents given q_i.

Ranking features typically can be expressed by query-dependent and query-independent relevance scores, being a natural split for distinct views, which inspires us to explore multi-view approach for rank learning. Nevertheless, possible decomposition of features is not limited to such an explicit split but can be optimized automatically, especially when feature definitions are unknown, which is not uncommon in the typical ranking dataset. Without loss of generality, we assume two views here for easy presentation.

3.1 View Consistency

Multi-view learning such as co-training [6] holds view consistency assumption that encourages the models on different views to give compatible predictions on the same instance with certain probability. In a ranking problem, given an input space $X = \{\mathbf{x}_i\}_{i=1}^{n} = \{(\mathbf{x}_i^1, \mathbf{x}_i^2)\}_{i=1}^{n}$, where $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$ correspond to the two views of the same list of documents of q_i (e.g., $\mathbf{x}_i^1 = \{\vec{x}_{ij}^1\}_{j=1}^{m_i}$ where $\vec{x}_{ij}^1$ is the first view of document ij), the view consistency assumption hypothesizes that for each q_i, all permutations of its documents with non-zero probability on $\mathbf{x}_i \in X$ agree with some target function on $\mathbf{x}_i^1$, i.e., $f_1 : \mathbf{x}_i^1 \rightarrow \pi_i$, and also agree with some target function on $\mathbf{x}_i^2$, i.e., $f_2 : \mathbf{x}_i^2 \rightarrow \pi_i$. However, this assumption seems too strong in most cases. In practice, some algorithms relaxed this assumption and tried to impose consensus constraint on the two view-based models directly with unlabeled examples [35, 19, 20, 28], which has been shown helpful to classification.

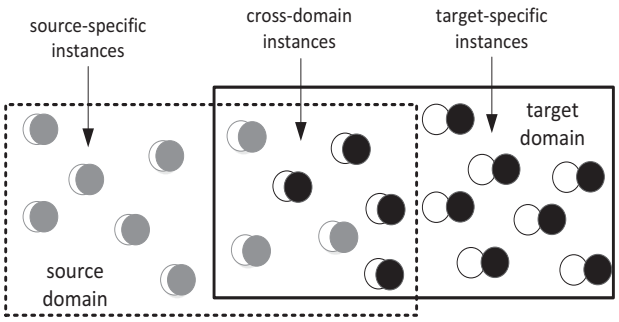


Figure 1: A conceptual illustration of view consistency under dataset shift

3.2 View Consistency under Dataset Shift

View consistency assumption would be more likely violated when training and test data come from different distributions or domains which is known as dataset shift [31]. Figure 1 conceptually illustrates the problem with a high-level representation of two related domains. In this figure, an instance[1] is represented using an hollowed circle and a filled circle coupled together, each corresponding to one view of the instance. The distance between the two coupled circles indicates the degree of agreement on that instance between the two models each learned from one view. The grey circle pairs are source domain instances and the dark pairs are of target domain. The instances intersecting two domains are called cross-domain instances that encode some general cross-domain commonality. And the other two regions that fall apart only encode specific knowledge of their own domains, referred to as domain-specific instances. Under dataset shift, the joint distribution of feature and relevance varies greatly between training and test data. Suppose we have two rankers each trained on one view using source training data. If we blindly apply them to target documents, their agreement on document rankings would be weakened, which is shown by the relatively larger distance between the coupled circles, due to the distribution gap of two domains.

Basically, the cross-domain instances from source domain would be helpful for learning target ranking model, and also, strengthening view consensus on target instances would be conducive to weighing out cross-domain source instances that are expected important to the success of adaptation, and vice versa.

[1]Note that in ranking, the instances may be documents, document pairs or queries depending on the specific scheme used. Here an instance corresponds to a query as we use listwise approach.

4. MULTI-VIEW RANKING APPROACH

Two views lacking of prediction consistency are subject to inferior ranking performance when they are combined. Our idea is to purposefully identify those important cross-domain instances from source data that can be used to boost target ranking performance by improving view consistency on target instances with a co-training-like procedure. As a result, the enhanced view consistency of target ranker can be expected helpful for selecting those informative cross-domain knowledge from the rest of source training data in a sense that the ranking of cross-domain source instances predicted by the target ranker would be better than that of source-specific instances. Therefore, some positive feedback would be resulted from the enhanced view consistency.

In this section, we describe the proposed multi-view ranking approach in semi-supervised as well as adaptation settings. First, we will introduce ListMLE, a state-of-the-art listwise ranking algorithm; Then, we will describe how it is extended and optimized by a multi-view approach.

4.1 ListMLE

Although listwise approach is natural and more effective, it is usually difficult to optimize a list-based loss directly because the performance measure on a document list is based on the result of a sort function. ListMLE employs a likelihood loss as the surrogate since it has nice properties [40]. Given the training data, it minimizes the sum of negative likelihood with respect to the ground-truth permutation of documents over all the training queries in $\{q_i\}_{i=1}^n$:

$$\min_{\vec{w}} \; L(\vec{w}) = \sum_{i=1}^{n} L_{q_i}(\vec{w}) = \sum_{i=1}^{n} -\log\left(P(\pi_{y_i}|\vec{w}, \mathbf{x}_i)\right) \qquad (1)$$

where $\vec{w}$ is the weight vector to estimate, $P(\pi_{y_i}|\vec{w}, \mathbf{x}_i)$ is the permutation probability based on the ground truth y_i of the documents of q_i, which is defined by Plackett-Luce model [30]:

$$P(\pi|\vec{w}, \mathbf{x}_i) = \prod_{j=1}^{m_i} \frac{\exp\left(f(\vec{w}, x_{\pi^{-1}(j)})\right)}{\sum_{k=j}^{m_i} \exp\left(f(\vec{w}, x_{\pi^{-1}(k)})\right)} \qquad (2)$$

where $\pi^{-1}(j)$ is the index of document ranked at the j-th position of permutation π, and $f(\cdot)$ is the specific expression of ranking function f.

The advantage of ListMLE also lies in its efficiency in training where one only needs to compute the probability of a single permutation based on ground truth but not all the permutations, and its objective function is easy to minimize using gradient-based optimizers.

4.2 Semi-supervised CoListMLE

Let $\vec{w}_1 = [w_l^1]_{l=1}^N$ and $\vec{w}_2 = [w_l^2]_{l=1}^N$ be the weight vectors of the ranking model for view $\mathbf{x}^1$ and $\mathbf{x}^2$, respectively, where N is the dimension of entire feature space, and $\mathbf{x}^1$ and $\mathbf{x}^2$ denote a non-overlapping separation of features according to $\vec{w}_1$ and $\vec{w}_2$. Let $Q_L = \{q_i\}_{i=1}^n$ and $Q_U = \{q_i\}_{i=1}^{n'}$ be the sets of (labeled) training queries and (unlabeled) test queries, respectively. Here we present a generalized semi-supervised framework for multi-view ranking called CoListMLE, inspired by the variant of co-training-based classification in [15].

Our objective is to minimize the combined likelihood rank loss of the two views on the labeled instances while maximizing the rank agreement between them on the unlabeled instances. Meanwhile, there are two critical issues for co-training to work: (1) the two view-based learners are trained on different subsets of features where each feature only can be used by one of the rankers; (2) the two view-based learners must be able to teach each other for boosting up learning performance, meaning that there should be sufficient number of unlabeled examples, on which only one of the learners can make confident prediction. The first condition constrains how to split the features, and the second was addressed by the ϵ-expanding rule in [3]. Our co-training-based ListMLE model named as CoListMLE is formulated as follows:

$$\min_{(\vec{w}_1, \vec{w}_2)} \; \sum_{i=1}^{n} \left[L_{q_i}(\vec{w}_1) + L_{q_i}(\vec{w}_2)\right] + \lambda \cdot D_{Q_U}(\vec{w}_1, \vec{w}_2) \qquad (3)$$

subject to:

(1) View splitting: $\sum_{l=1}^{N} \left[w_l^1 \cdot w_l^2\right]^2 = 0$

(2) ϵ-expandability:

$$\sum_{i=1}^{n'} \left[c_{\vec{w}_1}(q_i)\bar{c}_{\vec{w}_2}(q_i) + \bar{c}_{\vec{w}_1}(q_i)c_{\vec{w}_2}(q_i)\right] \geq \epsilon \min\left[\sum_{i=1}^{n'} \left[c_{\vec{w}_1}(q_i)c_{\vec{w}_2}(q_i)\right], \sum_{i=1}^{n'} \left[\bar{c}_{\vec{w}_1}(q_i)\bar{c}_{\vec{w}_2}(q_i)\right]\right]$$

Note that different from [15, 14], the first term of Equation 3 is the sum of likelihood-based rank loss of two views over all the labeled data, λ is a control coefficient, and the second term $D_{Q_U}(\cdot,\cdot)$ is newly introduced to capture listwise view disagreement estimated by the difference between two permutation probability distributions over all the unlabeled instances:

$$D_{Q_U}(\vec{w}_1, \vec{w}_2) = \sum_{i=1}^{n'} \Big[\log\left(P(\pi_i^{(k-1)}|\vec{w}_1, \mathbf{x}_i^1)\right) - \log\left(P(\pi_i^{(k-1)}|\vec{w}_2, \mathbf{x}_i^2)\right)\Big]^2 \qquad (4)$$

where $\pi_i^{(k-1)}$ is the permutation output by the model to the documents of q_i during the $(k-1)$-th round of iteration. The reason we have to use the previous permutation is that the permutation of current (i.e., the k-th) round is the function of $\vec{w}_1$ and $\vec{w}_2$, which are yet to be computed. Though enhancing view consistency is straightforward in classification, it remains challenging to do so for view-based rankers. It is hard to optimize the correlation between two predicted permutations directly in the objective due to the position-based, non-continuous and non-differentiable nature of most IR evaluation measures. Thus, we use the discrepancy between two permutation probabilities here as a surrogate.

The constraint (1) of Equation 3 controls the decomposition of features to be non-overlapping. In constraint (2) as to ϵ-expandability, $c_{\vec{w}_1}(q_i)$ or $c_{\vec{w}_2}(q_i)$ is a confidence indicator function of the corresponding view-based ranker regarding the prediction on query q_i, which is defined as:

$$c_{\vec{w}_1}^{(k)}(q_i) = \begin{cases} 1, & \text{if } \; P(\pi_i^{(k-1)}|\vec{w}_1, \mathbf{x}_i) > \tau; \\ 0, & \text{otherwise} \end{cases} \qquad (5)$$

where probability threshold τ can be set empirically. Also, $\bar{c}(q) = 1 - c(q)$ indicates that the ranker is not confident about q. The constraint (2) ensures that the number of queries in Q_U that exactly one ranker is confident about is

larger than the the number of queries which both rankers are confident about or both are not, so that both of the ranks can utilize those instances they are not confident about to learn from the other ranker that is confident on them [15].

This generalized framework allows us to use different ranking models and train the model in a semi-supervised fashion with automatically splitted views. Such objective function is non-convex [14] and non-smooth. We smooth the constraints using approximation and resort to augmented Lagrangian method [4] to transform it into an unconstrained problem[2]. Then locally optimal solution is obtained using the standard conjugate gradient descent.

4.3 AdaCoListMLE for Rank Adaptation

We assume a typical unsupervised adaptation setting, where all rank labels exist in source data, and there is no labeled data but reasonable amount of unlabeled data in target domain. Rank adaptation requires finding the most informative cross-domain knowledge from source domain as training materials to guide the transfer of ranking knowledge. As discussed in Section 3.2, the cross-domain source instances can carry some important commonality knowledge related to the target domain. Such kind of information will be identified for use to train the target ranker. If identified, presumably it can be leveraged to strengthen view consistency on target instances. Once the ranking on target domain is getting improved, the ranker becomes more target-leaning, then it will have more discriminant power to differentiate the cross-domain instances from source-specific ones. Therefore, we come up with a reinforcement process where the weighting of source queries and the degree of view consistency on target queries will be mutually enhanced in a unified framework, which is referred to as AdaCoListMLE:

$$\min_{(\vec{w}_1,\vec{w}_2)} \left\{ \sum_{i=1}^{n_s} W_{q_i} \cdot [L_{q_i}(\vec{w}_1) + L_{q_i}(\vec{w}_2)] \quad + \right. \tag{6}$$

$$\left. \lambda \cdot \sum_{i=1}^{n_t} \left[\log\left(P(\pi_i^{(k-1)}|\vec{w}_1, \mathbf{x}_i^1)\right) - \log\left(P(\pi_i^{(k-1)}|\vec{w}_2, \mathbf{x}_i^2)\right) \right]^2 \right\}$$

where W_{q_i} is the importance weight of source query q_i, and n_s and n_t are the number of source and target queries, respectively, and the same constraints of view splitting and ϵ-expandability are imposed like in Equation 3.

4.3.1 Source query weighting

Now the problem is how to weigh out the cross-domain source queries based on the ranking model obtained thus far. The intuition is that if the retrieved documents of a source query can be ranked well by a target ranking model, it implies that the rank knowledge of this query is more suitable for the target ranking task, thus should be deemed as more informative. Otherwise, this source training query may be not a good choice. Therefore, the importance of a source query can be measured by the performance of target ranking model on this query. We weigh a source training query q by directly using Normalized Discounted Cumulative Gain (NDCG) [26] obtained on it:

$$W_q = \frac{1}{Z_m} \sum_{j=1}^{m} \frac{2^{r(j)} - 1}{\log(1+j)} \tag{7}$$

[2]The derivation is not difficult which is omitted here due to space limit

Algorithm 1 AdaCoListMLE: iterative reinforcement rank adaptation algorithm based on CoListMLE

Input:
Q_s: Training queries in source domain, $|Q_s| > 0$
Q_t: Unlabeled queries in target domain, $|Q_t| > 0$
λ, ϵ, τ, κ: The free parameters

Output:
Target ranking model M;

1: Initialize the weights of queries in Q_s as 1;
2: Initialize the permutation output $\pi^{(0)}$ using ListMLE trained on Q_s;
3: **repeat**
4: Solve optimization problem in Equation 6 and obtain ranking model M with weight vectors $\vec{w}_1$, $\vec{w}_2$;
5: Use function $f = (\vec{w}_1 + \vec{w}_2) \cdot \mathbf{x}$ to rank the documents of target queries in Q_t;
6: Move κ most confident target queries from Q_t to Q_s;
7: Train a target ranking model M' with the predicted target data in step 5;
8: Use M' to re-weigh the source queries using Equation 7;
9: **until** Converged (no more prediction is confident);
10: **return** Target ranking model M;

where $r(j)$ denotes the rank label of the j-th document in the ranked list of q, m is the m-th top position in the list and Z_m is a normalization constant which is chosen so that the perfect list gets score of 1. Here we adopt NDCG@10.

4.3.2 Reinforcement training procedure

The iterative reinforcement procedure for query weighting and view consistency enhancement is given in Algorithm 1. Step 1-2 initializes the weights of source training queries and the permutation output; Step 4 optimizes the objective function to obtain the two view-based rankers; Step 5 ranks the documents of target queries using the model learned thus far; Step 6 moves some most confident target queries to the source training set so that the model can be re-trained with the expanded dataset in next iteration; Step 7 trains a different ranker using the predicted target data in step 6, which is used to re-weigh the source queries using Equation 7 in step 8. The iteration continues until convergence is reached.

Note that step 7 trains target ranking model M' using the predicted target data independent of current model M for re-weighing source queries. Actually, another seemingly straightforward solution is to re-weigh the source queries by letting M, which is trained using the mixture of source training queries and most-confidently predicted target queries, rank the source documents. Both solutions have pros and cons: M' uses pure target data but is trained with the ranks predicted by M which may not be sufficiently accurate; M is trained with more accurate ranks, but most of the training examples are from source domain, rendering its prediction on source queries source-leaning, and moreover, moving around queries from Q_t to Q_s in step 6 may not always lead to improved effectiveness as the confidence of ranker depends on how different two domains are at large. Therefore, using M to re-weigh source queries is liable to some obvious risk. On the other hand, the predictive power of M on target data may directly reflect the strengthened view agreement. Using the predicted target queries to train M' will make sense provided that certain target ranking accuracy by M could

be reached. In fact, we conducted experiments using both of the solutions and found that source query weighting using M' consistently outperformed using M. Therefore, we adopted M' for re-weighing source queries.

5. EXPERIMENTS AND RESULTS

Since semi-supervised ranking is a special case of rank adaptation, we put emphasis on adaptation experiments. In-domain semi-supervised experiments are only conducted on one of the data sets.

5.1 Data Sets

We used two data sets for experiments: LETOR3.0[3] and Yahoo Learning to Ranking Challenge[4] data sets.

LETOR3.0 was constructed from the collections of TREC 2003 and 2004 Web track. The raw data were preprocessed into the standard format for learning to rank [33]. Each query-document pair is represented as 64 features, including both low-level features such as term frequency, inverse document frequency and document length, and high-level features such as BM25, language-modeling, PageRank and HITS. The relevance judgments take binary relevance values. Three query tasks were defined, namely Home Page Finding (HP), Named Page Finding (NP) and Topic Distillation (TD) [38]. HP aims to return the homepage of the specific organization or person. NP is required to return the page whose name is exactly the query. TD is to return a list of entry points of good websites that contain the contents relevant to the topic, and its focus is to return entry pages of good websites rather than the pages containing the relevant contents themselves. We can treat each query task as a separate domain like [12, 23, 22], which is only available in the 3.0 edition of LETOR data.

Yahoo dataset provides two sets of rank training data of search engines each from a different country and language. The original data include a large set (Set1) with 29,921 queries and a small set (Set2) with 6,330 queries. Each dataset is divided into 3 sets for training, validation and test. There are 700 features in this data, but their definition is not provided explicitly. Some features are defined in source or target domain only while some others are defined in both domains. We simply expanded the feature space to include all features of both domains where the missing features were put as 0. The relevance judgment takes 5 different levels from 0 (irrelevant) to 4 (perfectly relevant).

Since HP and NP are rather similar tasks but TD is different, we conducted experiments across TD and the other two tasks to examine adaptation performance. For Yahoo dataset, we use the training data of Set1 as source domain, and the test data of Set2 as target domain. We do not use any relevance label of target data during adaptation training. We also conducted in-domain training by using the training part of Set2 to examine single domain ranking performance. Table 1 shows statistics of the data sets used.

5.2 Setup

Our multi-view algorithms are compared with three *single-view* baselines: (1) `naiveList`: A ListMLE trained in source domain is naively applied to target domain; (2) `dsList`: ListMLE that is trained by using domain-separator-based weighting method described in [22] for source query weighting, which is state-of-the-art rank adaptation based on instance weighting; (3) `wList`: This is a variant of `dsList` using our proposed query weighting technique, i.e., the weights of source queries are updated iteratively with Equation 7 by the ranker trained on the predicted target queries. It is also a single-view version of AdaCoListMLE descried in Algorithm 1.

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

[4] `http://learningtorankchallenge.yahoo.com/datasets.php`

Table 1: Statistics of the two data sets with different domains used for experiments

Data set	Domains	# queries	# docs/query
LETOR	NP04	75	1,000
	HP04	75	1,000
	TD04	75	1,000
	TD03	50	1,000
Yahoo	Set1-train	19,944	27.5
	Set2-train	1,266	27.5
	Set2-test	3,798	27.2

The proposed multi-view algorithms are examined under four configurations: (1) `CoList` ($\lambda = 0$): CoListMLE with no view-consistency term in Equation 3, which equals to the ranking version of [15]; (2) `CoList`: CoListMLE with consistency term imposed in Equation 3; (3) `AdaCoList` ($\lambda = 0$): AdaCoListMLE with no view-consistency term in Equation 6 (but with query weighting); (4) `AdaCoList`: AdaCoListMLE with consistency term imposed in Equation 6 (i.e., with both query weighting and view consistency).

The free parameter λ was tuned using the validation set of source data. We empirically set parameters $\epsilon = 1.0$, $\tau = 0.8$, and $\kappa = 5\%$.

The performance was measured by NDCG@1,3,5,10 [26] and Expected Reciprocal Rank (ERR) [10]. NDCG of a single query was given by Equation 7. ERR of a query is computed as $ERR = \sum_{i=1}^{m} \frac{1}{i} R(r_i) \prod_{j=1}^{i-1} (1 - R(r_j))$, where r_i is the relevance level of document ranked at the i-th position, $R(r) = \frac{2^r - 1}{2^{r_{max}}}$ is a mapping from relevance level to probability of relevance (r_{max} is the maximum relevance level), and m is the length of the ranked list. We then take the average value of the measure over all test queries.

5.3 Results and Discussions

5.3.1 *Effectiveness of view decomposition*

The principle of multi-view algorithms is that the redundancy and complementarity between different decompositions of features help view-based learners teach each other so as to boost up overall learning performance. Therefore, it is necessary to show that this basic hypothesis holds for ranking by examining how the view decomposition can exert influence to ranking compared to a single-view baseline and different decomposition schemes.

We use `naiveList` as baseline and compare to `CoList` ($\lambda = 0$) using two different splitting methods: (1) natural split (`natural`), which exploits explicit views, i.e., query-dependent and query-independent features. For this, we resort to the feature definition given in LETOR dataset [33], where 52 features in the categories "Q" and "Q-D" are considered as query-dependent, and 12 features of category "D"

Table 2: The effectiveness of different view decomposition compared with single-view baseline: `naive` – NaiveListMLE; `natural` – Natural split; `auto` – Automatic split. The best performance is bolded, and † indicates significantly better than `naive` at 95% confidence level based on paired two-tailed t-test

	Methods	TD04-HP04	TD04-NP04	TD03-HP04
NDCG1	**naive**	0.356	0.289	0.453
	natural	0.222	0.333	0.444
	auto	**0.422**	**0.467**†	**0.471**
NDCG3	naive	0.539	0.562	0.547
	natural	0.567	0.479	0.572
	auto	**0.642**†	**0.620**†	**0.584**†
NDCG5	naive	0.561	0.603	0.582
	natural	0.601	0.519	0.606
	auto	**0.668**†	**0.630**	**0.622**†
NDCG10	naive	0.599	0.641	0.630
	natural	0.625	0.561	0.642
	auto	**0.692**†	**0.686**†	**0.662**†
ERR	naive	0.247	0.237	0.280
	natural	0.221	0.230	0.282
	auto	**0.292**†	**0.293**†	**0.293**†

as query-independent; (2) automatic split (`auto`), which resorts to the constraint (1) for automatic feature decomposition (see Equation 3). For fair comparisons, we just make CoListMLE perform in an unsupervised adaptation fashion and let two views trained with only one iteration of co-training not allowing further model updates. This aims to show whether feature decomposition is advantageous over the usual single-view treatment.

As shown in Table 2, the automatic split always outperforms `naive` and in most cases `auto` is significantly better, indicating that proper feature decomposition is in general more advantageous. The explicit views based on query dependency do not seem to be an effective feature decomposition scheme since the performance of `natural` varies greatly and is not significantly different from that of `naive`. This implies that using optimization techniques for good decomposition is crucial. Also, this is not the only reason why we use automatic split. Oftentimes, there is no specific feature definition provided in ranking benchmark datasets, in which case one has to decompose features in a principled way to work with multi-view ranker.

5.3.2 *Effectiveness of rank adaptation on LETOR*

In this experiment, we compare our approach with single-view and multi-view baselines using LETOR dataset. We conducted cross-domain ranking in two adaptation settings: TD04-NP04 and TD04-HP04. Figure 2 shows the performance. Paired two-tailed t-test results are detailed in Table 3, where we show p-values between `AdaCoList` and all other methods, and also p-values between the basic multi-view configuration `CoList` ($\lambda = 0$) and other multi-view configurations.

We have the following findings and discussions based on these results:

— Multi-view rankers generally performed better than traditional single-view rankers, indicating that splitting ranking features into views is the right direction for ranking. What really unleash the power of different views is the addition of view consistency factor imposed in our objective. We find that the performance clearly worsens when view consistency term is removed, i.e., λ is set to zero in `CoList` and `AdaCoList`. This suggests that strengthening the agreement of view-based rankers on target queries is the key to the success of rank adaptation. Such an improvement owes to the cooperation between the ϵ-expandability constraint and the view consistency term: The ϵ-expandability ensures to capture those informative target instances, with which two view-based rankers can learn from each other. As a result, their ranking consensus could be gradually improved and explicitly captured by the view consistency term.

— Another crucial factor to adaptation is the appropriate weighting of source queries. Our weighting scheme is simple yet very effective, evidenced by the improvement of ranking not only when used in multi-view case but also in the single-view approach `wList`. `wList` always outperforms `dsList` in large margin, while the latter does not appear robust and sometimes is even worse than `naiveList`. Weighting scheme used by `dsList` is based on the so-called domain separator approach where the weight values are derived from the classification hyperplane of two domains, which although representing domain relatedness in some extent, cannot reflect the importance of source training queries in terms of ranking performance measure like our weighting scheme.

— Clearly, the two factors, one for strengthening view consistency on target queries and the other for appropriate weighting of source queries, can mutually reinforce each other in our proposed framework. This is reflected by the fact that `AdaCoList` significantly outperforms `CoList` and `AdaCoList` ($\lambda = 0$) in accordance with most of performance measures. The reason is straightforward: enhanced view agreement on target queries will lead to a better target-leaning ranker, which in return can help weigh out those cross-domain source queries since they receive higher ranking scores (in terms of NDCG@10) predicted by model M'.

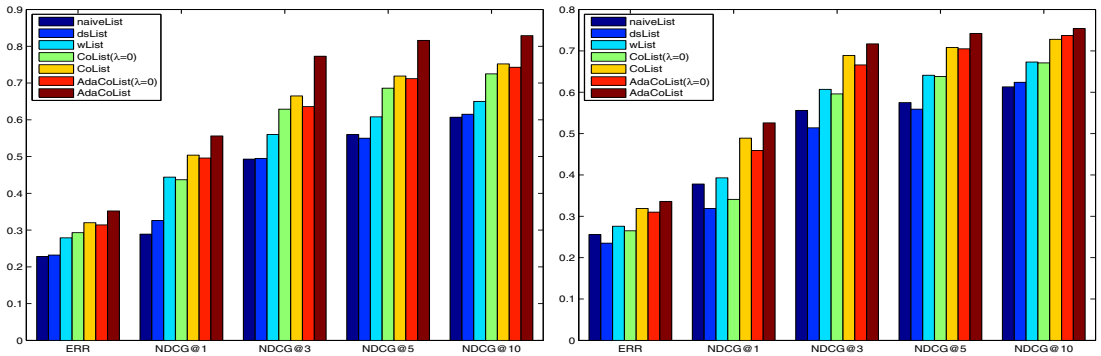
— As shown in the left bottom part of Table 3, only imposing view consistency or only using source query weighting cannot ensure significant improvement over the multi-view baseline `CoList` ($\lambda = 0$). For example, on TD04-NP04, the NDCG@3,5,10 obtained may not necessarily significantly different. But combining the two components in the ultimate model makes them effectively reinforced.

5.3.3 *Effectiveness of rank adaptation on Yahoo data*

We conducted adaptation experiments on Yahoo dataset. Based on `AdaCoList`, we studied the sensitivity of coefficient λ that is used to control the extent of view consistency in the model, for which we examined different values of λ to show how ERR changed with it. Meanwhile, we also studied the trend of ranking performance varying with iteration. Both results are displayed in Figure 3[5].

We observe that large λ typically results in poor adaptation performance, and λ between 0 and 1 can give reasonably good results, and the best is achieved when $\lambda = 1$. This implies that the view consistency factor should be carefully chosen to avoid its side effect: (1) if weak view consistency is imposed due to very small λ, the magnitude of consistency term would be too weak to influence the loss to lean towards target direction; (2) if its magnitude is very large such as when $\lambda = 1000$, the consistency part will dominate the entire rank loss, making the model hard to keep certain level of accuracy, which means the two views being frequently and consistently wrong. Therefore, setting an appropriate value to λ is helpful. But overall, the extent of performance vari-

[5]We stopped training after 10 iterations in the cases of $\lambda > 1$ due to their obviously poor performance



(a) TD04-NP04 adaptation results (b) TD04-HP04 adaptation resutls

Figure 2: The comparison of different algorithms for ranking adaptation using LETOR3.0 dataset. The source domain is Topic Distillation (TD) task in TREC-2004 (TD04) and the target domains are Named Page finding task (NP04) and Home Page finding task (HP04). `naiveList`: A single-view ListMLE ranker trained on source domain is naively applied to target domain; `dsList`: A single-view semi-supervised ListMLE using domain-separator based source query weighting described in [22]; `wList`: A variant of `dsList` using our proposed query weighting technique and equal to single-view version of AdaCoListMLE; `CoList` ($\lambda = 0$): CoListMLE w/o view consistency (i.e., ranking version of [15]); `CoList`: CoListMLE with view consistency; `AdaCoList` ($\lambda = 0$): AdaCoListMLE w/o view consistency; `AdaCoList`: AdaCoListMLE with view consistency

Table 3: Results of paired two-tailed t-test (p-values) between different algorithms and configurations

	TD04-NP04					TD04-HP04				
	NDCG1	NDCG3	NDCG5	NDCG10	ERR	NDCG1	NDCG3	NDCG5	NDCG10	ERR
	`AdaCoList`									
`naiveList`	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	0.03	0.01	< 0.01	< 0.01	< 0.01
`dsList`	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	< 0.01	$\ll 0.01$	$\ll 0.01$	< 0.01	$\ll 0.01$
`wList`	0.04	$\ll 0.01$	$\ll 0.01$	$\ll 0.01$	< 0.01	0.02	0.03	0.03	< 0.05	< 0.01
`CoList` ($\lambda = 0$)	0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	$\ll 0.01$
`CoList`	0.09	< 0.01	< 0.01	< 0.01	0.01	0.13	0.11	0.04	0.04	< 0.05
`AdaCoList` ($\lambda = 0$)	0.10	< 0.01	< 0.01	< 0.01	0.02	0.07	0.09	0.05	0.05	0.04
	`CoList` ($\lambda = 0$)									
`CoList`	0.02	0.11	0.13	0.08	0.01	0.01	< 0.01	0.01	0.02	< 0.01
`AdaCoList` ($\lambda = 0$)	0.03	0.34	0.07	< 0.05	< 0.01	0.01	0.01	< 0.01	$\ll 0.01$	$\ll 0.01$
`AdaCoList`	0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	$\ll 0.01$

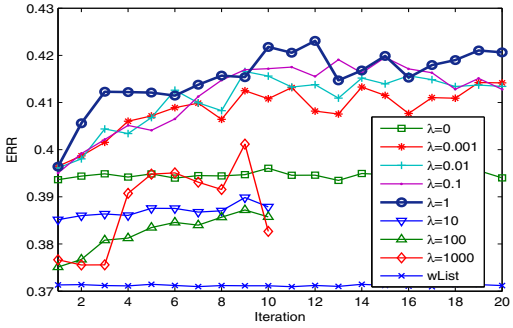


Figure 3: Rank adaptation performance of `AdaCoList` varies with iterations and λ based on Yahoo dataset

ation, being within a magnitude of 0.1, can be considered not sensitive to λ.

Typically, iteration improves adaptation when $\lambda > 0$, indicating that view consistency and source query weighting are mutually strengthened. The curve remains flat when $\lambda = 0$, meaning that source query weighting does not work alone if without the view consistency imposed on target data. This is because such a case forms some pointless loop, where the algorithm iterates between a purely source-trained ranker and a target ranker trained with inaccurate rank predictions, and there is no any boosting point. Note that unlike on LETOR data, here we skip performing step 6 in Algorithm 1 because it was found that the move of queries worsened performance gradually. This may be due to the large domain gap of Yahoo data that came from search engines of different languages [11], and inappropriately using target queries may introduce considerable noise into source domain. Therefore, no target queries are selected into source domain during iteration. And we yet did not find reliable way to select these κ target queries. We will leave it for future work.

The comparison of rank adaptation performance on Yahoo dataset is shown in Figure 4. We can see the similar trend of performance gain as using LETOR data. T-test indicates that `AdaCoList` is significantly better than all other methods and configurations (p-value<0.01). This again justifies our method is effective.

A different trend is that `wList` is not obviously better than other single-view baselines. This is simply because our weighting method has to rely on the improvement of the predictive power of current ranking model to give a better

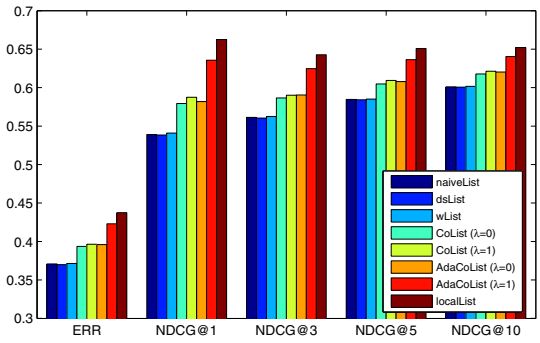


Figure 4: Rank adaptation performance on Yahoo dataset, where `localList` is the performance of ListMLE trained locally on target domain

Table 4: The comparison of ranking performance in single domain based on Yahoo Set2. * indicates significantly better than baseline ListMLE at 99% confidence level according to paired two-tailed t-test

Metric	ListMLE	CoList (λ=0)	CoList (λ=1)
ERR	0.437	**0.445***	0.440*
NDCG@1	0.663	**0.684***	0.662
NDCG@3	0.643	**0.660***	0.646
NDCG@5	0.651	**0.665***	0.656*
NDCG@10	0.652	**0.664***	0.657*

ranking prediction on target data. As discussed previously, because no target query was moved to source domain, there is no any boosting point for improving the current model as shown by the flat curve of `wList` also in Figure 3. Similarly, we found that introducing target queries to source domain based on confidence is harmful on this dataset, also probably due to the large distribution gap between two domains as described in [11].

Two strategies may improve the case. One is to exploit the presented multi-view approach by incorporating view consistency to work together with query weights which has demonstrated salient gains as shown in Figure 4. The other is to consider some reliable way for selecting and moving those target queries really useful to source domain in our future work.

5.3.4 *In-domain ranking performance on Yahoo data*

We also studied the performance of the proposed multi-view approach for ranking in a single domain of Yahoo data. We trained the model using Set2-train and tested it on the Set2-test. The results are displayed in Table 4.

It can be observed that the two multi-view rankers both outperform the baseline ListMLE. T-test shows that `CoList` ($\lambda = 0$) is significantly better in terms of all the performance measures (p-value<0.01), and `CoList` is significantly better based on ERR and NDCG@5,10 (p-value<0.01). This verifies the effectiveness of multi-view approach for ranking in a single domain.

It is noted that in contrast to the performance in adaptation setting, using view consistency term worsens the in-domain ranking performance a little than not considering consistency-based loss. However, there is no significant difference between these two variants of `CoList`. The reason of performance drop is that when consistency-based loss on the data from same distribution is added, it will necessarily dilute the rank loss portion of the objective function, which may result in some degradation of ranking accuracy. Therefore, enhancing view consistency is less important for in-domain ranking. But it seems vital for adaptation since improving target ranking accuracy relies on effective query weighting which can benefit from enhanced view consistency.

6. CONCLUSION AND FUTURE WORK

We present a novel and effective ranking framework based on the principle of multi-view learning and listwise ranking approach. Our method is motivated by the description of ranking features from different perspectives which is conformant to the nature of multi-view learning paradigm. The idea is to strengthen the view consistency of rankings on the unlabeled data with a co-training-like procedure where the two view-based rankers can iteratively boost each other. Then our method is generalized to the typical rank adaptation scenario where training and test data follow different distributions and no target training data is available. To overcome the lack of view consensus due to distribution gap, it aims to mutually reinforce the view consistency of ranking for target queries and the weighting of source queries. The rationale is that imposing view consistency on target queries improves the target ranker which in return can help weigh out those cross-domain source queries that can be used to further strengthen the view consistency. Our method results in significant improvement over some strong single-view and multi-view baselines by rank adaptation on LETOR and Yahoo learning to rank datasets, and is also shown effective for in-domain rank learning on Yahoo dataset.

Our framework is generic that can accommodate various ranking and adaptation approaches. Existing single-view ranking algorithms can be upgraded using our multi-view scheme with reasonably small amount of effort. Also, for scalability aspect, extending our method to more than two views will be fairly straightforward, for example, by incorporating more pairwise consistency terms in the objective function resulting from the additional views.

There are a few more interesting directions to follow in our future work: we plan to investigate reliable ways of moving useful target queries to source domain to improve adaptation; we can try different ranking schemes and rank correlation measures in objective function; there also remain interesting theoretical issues regarding this topic worth of further study.

7. REFERENCES

[1] S. Abney. Bootstrapping. In *ACL*, pp. 360–367, 2002.
[2] M.-R. Amini, T.-V. Truong, and C. Goutte. A boosting algorithm for learning bipartite ranking functions with partially labeled data. In *SIGIR*, pp. 99–106, 2008.
[3] M. F. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. In *NIPS*, pp. 87–96, 2004.
[4] D. P. Bertsekas, W. W. Hager, and O. L. Mangasarian. *Nonlinear programming*. Athena Scientic Belmont, MA, 1999.

[5] J. Blitzer, R. Mcdonald, and F. Pereira. Domain adapation with structural correspondce learning. In *EMNLP*, pp. 120–128, 2006.

[6] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pp. 92–100, 1998.

[7] U. Brefeld and T. Scheffer. Co-EM support vector learning. In *ICML*, pp. 121–128, 2004.

[8] P. Cai, W. Gao, A. Zhou, and K.-F. Wong. Relevant knowledge helps in choosing right teacher: Active query selection for ranking adaptation. In *SIGIR*, pp. 115–124, 2011.

[9] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *ICML*, pp. 129–136, 2007.

[10] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*, pp. 621–630, 2009.

[11] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *JMLR Workshop and Conference Proceedings*, 14:1–24, 2011.

[12] D. Chen, Y. Xiong, J. Yan, G.-R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253, 2009.

[13] K. Chen, R. Lu, C. Wong, G. Sun, L. Heck, and B. Tseng. Trada: Tree based ranking function adaptation. In *CIKM*, pp. 1143–1152, 2008.

[14] M. Chen, K. Q. Weinberger, and J. Blitzer. Co-training for domain adaptation. In *NIPS*, pp. 271–278, 2011.

[15] M. Chen, K. Q. Weinberger, and Y. Chen. Automatic feature decomposition for single view co-training. In *ICML*, pp. 953–960, 2011.

[16] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *EMNLP*, pp. 100–110, 1999.

[17] S. Dasgupta, M. L. Littman, and D. McAllester. PAC generalization bounds for co-training. In *NIPS*, pp. 375–382, 2001.

[18] K. Duh and K. Kirchhoff. Learning to rank with partially-labeled data. In *SIGIR*, pp. 251–258, 2008.

[19] J. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmak. Two view learning: SVM-2K, theory and practice. In *NIPS*, 2006.

[20] K. Ganchev, J. Graça, J. Blitzer, and B. Taskar. Multi-view learning over structured and non-identical outputs. In *UAI*, pp. 204–211, 2008.

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

[22] W. Gao, P. Cai, K.-F. Wong, and A. Zhou. Learning to rank only using training data from related domain. In *SIGIR*, pp. 162–169, 2010.

[23] B. Geng, L. Yang, C. Xu, and X.-S. Hua. Ranking model adaptation for domain-specific search. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):745–758, 2010.

[24] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, pp. 256–263, 2007.

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

[26] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pp. 41–48, 2000.

[27] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, pp. 264–271, 2007.

[28] G. Li, S. C. H. Hoi, and K. Chang. Two-view transductive support vector machines. In *SIAM SDM*, pp. 235–244, 2010.

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

[30] J. Marden. *Analyzing and Modeling Rank Data*. Champman and Hall/CRC, 1995.

[31] J. Q. nero Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence. *Dataset Shift in Machine Learning*. MIT Press, Cambridge, 2009.

[32] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pp. 86–93, 2000.

[33] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

[34] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li. Query-level loss functions for information retrieval. *Information Processing and Management*, 44:838–855, 2008.

[35] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. In *ICML Workshop on Learning with Multiple Views*, 2005.

[36] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: Optimising non-smooth rank metrics. In *WSDM*, pp. 77–86, 2008.

[37] N. Usunier, M.-R. Amini, and C. Goutte. Multiview semi-supervised learning for ranking multilingual documents. In *ECML-PKDD*, pp. 443–458, 2011.

[38] E. M. Voorhees. Overview of TREC 2004. In *TREC*, pp. 1–12, 2004.

[39] B. Wang, J. Tang, W. Fan, S. Chen, Z. Yang, and Y. Liu. Heterogeneous cross domain ranking in latent space. In *CIKM*, pp. 987–996, 2009.

[40] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank – theory and algorithm. In *ICML*, pp. 1192–1199, 2008.

[41] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pp. 189–196, 1995.

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

[43] B. Zadronzny. Learning and evaluating classifiers under sample selection bias. In *ICML*, pp. 114–121, 2004.

[44] D. Zhang, J. He, Y. Liu, L. Si, and R. D. Lawrence. Multi-view transfer learning with a large margin approach. In *SIGKDD*, pp. 1208–1216, 2011.