Ranking with Auxiliary Data

Bo Long
Yahoo! Labs
701 First Avenue
Sunnyvale, California 94089
bolong@yahoo-inc.com

Yi Chang
Yahoo! Labs
701 First Avenue
Sunnyvale, California 94089
yichang@yahoo-inc.com

Srinivas Vadrevu
Yahoo! Labs
701 First Avenue
Sunnyvale, California 94089
svadrevu@yahoo-inc.com

Shuang Hong Yang
College of Computing
Georgia Tech
shy@gatech.edu

Zhaohui Zheng
Yahoo! Labs
701 First Avenue
Sunnyvale, California 94089
zhaohui@yahoo-inc.com

ABSTRACT

Learning to rank arises in many information retrieval applications, ranging from Web search engine, online advertising to recommendation system. In learning to rank, the performance of a ranking function heavily depends on the number of labeled examples in the training set; on the other hand, obtaining labeled examples for training data is very expensive and time-consuming. This presents a great need for making use of available auxiliary data, i.e., the within-domain unlabeled data and the out-of-domain labeled data. In this paper, we propose a general framework for ranking with auxiliary data, which is applicable to various ranking applications. Under this framework, we derive a generic ranking algorithm to effectively make use of both the within-domain unlabeled data and the out-of-domain labeled data. The proposed algorithm iteratively learns ranking functions for target domain and source domains and enforces their consensus on the unlabeled data in the target domain.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.5.1 [**Pattern Recognition**]: Models-statistical

General Terms

algorithms

Keywords

Ranking, Auxiliary Data, Transfer learning, Semi-supervised learning, Gradient boosting, Source Domain, Target Domain

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

1. INTRODUCTION

Ranking is the core component of many important information retrieval problems, such as web search, recommendation, computational advertising. Learning to rank represents an important class of supervised machine learning tasks with the goal of automatically constructing ranking functions from training data. As many other supervised machine learning problems, the quality of a ranking function is highly correlated with the amount of labeled data used to train the function. Due to the complexity of many ranking problems, a large amount of labeled training examples are usually required to learn a high quality ranking function. However, in most applications, while it is easy to collect unlabeled samples, it is very expensive and time-consuming to acquire labeled data.

Hence, ranking with auxiliary data comes as a paradigm to reduce the labeling effort in learning to rank. In general, auxiliary data can be any data except the labeled data from the target domain. In this study, we use auxiliary data to denote the following two types of data. The first one is the out-of-domain labeled data, i.e., the labeled data from the related domains. The second is the within-domain unlabeled data, i.e., the unlabeled data from the target domain itself.

In this paper, we propose a general model to leverage both out-of-domain labeled data and within-domain unlabeled data for learning to rank. By optimizing the consensus between the target domain ranking function and the source domain ranking functions on the unlabeled data from the target domain, the proposed model is capable of (1) transferring consistent information into the final target ranking function to improve the robustness and (2) adapting the target ranking function to the real-application data distribution to reduce the overfilling. Our main contributions can summarized as follows.

2. MODEL AND ALGORITHM

Let us first consider learning to rank under traditional empirical risk minimization framework. Suppose that we only have training data from the target domain, $\mathcal{T} = \{(a_i, b_i)|a_i \succ$

$b_i\}_{i=1}^n$, where $a_i, b_i \in \mathbb{R}^d$ denote the feature vectors for query-document examples and $a_i \succ b_i$ denotes that a_i is preferred over b_i, i.e., a_i should be ranked higher than b_i. Without loss of generality, here we assume that the input is in the form of pairwise preference, since absolute relevant judgement data and listwise preference data can be easily transformed into the pairwise preference data.

Then, we formulate the problem of learning ranking functions as computing a ranking function $h \in \mathcal{H}$, where $\mathcal{H}$ is a given function class, such that h match the set of preferences, i.e., $h(a_i) > h(b_i)$, if $a_i \succ b_i; i = 1, 2, ..., n$, as much as possible. Given a loss function l to measure the preference match, we have the following empirical risk objective function,

$$\tilde{\mathcal{R}}(h) = \frac{1}{n}\sum_{i=1}^{n} l(h(a_i), h(b_i)) \tag{1}$$

Under the conventional risk minimization, if the training data sample size n is small, the empirical risk $\tilde{\mathcal{R}}(h)$ is not a reliable estimate of the true risk $\mathcal{R}(h)$ and we cannot expect to obtain a good estimation of the target function h.

In the ranking with auxiliary data setting, besides a small set of labeled data from the target domain $\mathcal{T}$, we have additional labeled data from m source domains,

$$\mathcal{S}^{(j)} = \{(a_i^{(j)}, b_i^{(j)}) | a_i^{(j)} \succ b_i^{(j)}\}_{i=1}^{n_j} \tag{2}$$

for $j = 1, 2, ..., m$, and a large amount of unlabeled data from the target domain, $\mathcal{U} = \{(c_i, d_i)\}_{i=1}^{n_u}$. For convenience, we still put the unlabeled data in the format of pairs, even we do not have the pairwise preference label information.

Now the goal is to use $\mathcal{S}^{(j)}$ and $\mathcal{U}^{(j)}$ to help learn the ranking function h. To achieve this goal, we need to incorporate useful information from $\mathcal{S}^{(j)}$ and $\mathcal{U}^{(j)}$ into h. For the out-of-domain data $\mathcal{S}^{(j)}$,it is difficult to directly measure how useful they are, since it is difficult to measure how difference between the true distributions of the source domains and the true distribution of the target domain. However, intuitively, if we have an optimal ranking function $h^{(j)}$ based on $\mathcal{U}^{(j)}$, then the performance of $h^{(j)}$ on the target domain will provide a reasonable indication of the usefulness of $\mathcal{S}^{(j)}$. For example, if a optimal ranking function from English Web search domain provides just a random guess of the ranking on another language domain, we cannot expect too much usefulness of the English domain for this target domain. This enlightens us that one way to extract useful information from source domains $\mathcal{S}^{(j)}$ is through the ranking function $h^{(j)}$. For the within-domain data, the most advantage is that they can provide a data distribution very close to the real application data to which the learned ranking function h will be applied to.

Based on the above observations, we propose the following model to learn the ranking function h with auxiliary data,

$$\begin{aligned}\tilde{\mathcal{R}}(h, \{h^{(j)}\}_{j=1}^{n_j}) &= \alpha\sum_{i=1}^{n} l(h(a_i), h(b_i)) + \\ &\sum_{j=1}^{m}\beta^{(j)}\sum_{i=1}^{n_j} l(h^{(j)}(a_i^{(j)}), h^{(j)}(b_i^{(j)})) + \\ &\gamma\sum_{j=1}^{m}\sum_{i=1}^{n_u} l(h(c_i), h(d_i), h^{(j)}(c_i), h^{(j)}(d_j)),\end{aligned} \tag{3}$$

where α, $\beta^{(j)}$, and γ are non-negative constants that denote weights for each components. The first component is based on the labeled data from the target domain $\mathcal{T}$; the second component is based on the labeled data from the source domain $\mathcal{S}^{(j)}$; the third component aims to enforce the target domain ranking function h and the source domain ranking functions $h^{(j)}$ to be "consensus" on the unlabeled data $\mathcal{U}$ under the loss function l (here we overload the notation l for simplicity). The intuition behind this model formulation is that if h and $h^{(j)}$ predict "consistent" ranking on the unlabeled examples that consist of a representative sample of real application data, then h is more likely predict correct ranking on the real application data, since h obtains more "confidence" in those real application examples during training process with the help from the source domain ranking functions $h^{(j)}$.

Another important issue is how to design the loss function l. We propose following loss functions for the ranking with auxiliary data model in (3).

For the labeled data $\mathcal{T}$ and $\mathcal{S}^{(j)}$, we use the following loss function,

$$l(h(a_i), h(b_i)) = \frac{1}{2}(max\{0, h(b_i) - h(a_i)\})^2. \tag{4}$$

The motivation is that for a labeled pair $a_i \succ b_i$, if h matches the given preference, i.e., $h(a_i) > h(b_i)$, then h incurs no cost on the pair, otherwise the cost is given by $(h(b_i) - h(a_i))^2$.

For the unlabeled data $\mathcal{U}$, the loss function l is to measure the consistence between h and $h^{(j)}$. We use the following function,

$$\begin{aligned}&l(h(c_i), h(d_i), h^{(j)}(c_i), h^{(j)}(d_i)) \\ &= \frac{1}{2}(max\{0, h(d_i) - h(c_i)\})^2\mathbf{I}(h^{(j)}(c_i) > h^{(j)}(d_i)) \\ &+\frac{1}{2}(max\{0, h(c_i) - h(d_i)\})^2\mathbf{I}(h^{(j)}(c_i) \leq h^{(j)}(d_i)),\end{aligned} \tag{5}$$

where $\mathbf{I}$ is an indicator function such that

$$\mathbf{I}(h^{(j)}(c_i) > h^{(j)}(d_i)) = \begin{cases}1 & \text{if } h^{(j)}(c_i) > h^{(j)}(d_i), \\ 0 & \text{else.}\end{cases} \tag{6}$$

For loss function in (5), if h and $h^{(j)}$ predict the same order of preference for the pair (c_i, d_i), the loss is 0; otherwise, the loss is given by $(h(c_i) - h(d_i))^2$.

Furthermore, to avoid constant solution, we revise the loss functions to include a margin,

$$l(h(a_i), h(b_i)) = \frac{1}{2}(max\{0, h(b_i) - h(a_i) + \tau\})^2, \tag{7}$$

and

$$l(h(c_i), h(d_i), h^{(j)}(c_i), h^{(j)}(d_i))$$
$$= \frac{1}{2}(max\{0, h(d_i) - h(c_i) + \tau\})^2 \mathbf{I}(h^{(j)}(c_i) > h^{(j)}(d_i))$$
$$+ \frac{1}{2}(max\{0, h(c_i) - h(d_i) + \tau\})^2 \mathbf{I}(h^{(j)}(c_i) \leq h^{(j)}(d_i)), \quad (8)$$

where $0 < \tau \leq 1$ is a constant.

Under the above model formulation, we we adopt a functional gradient boost approach [2] to derive a gradient boosting algorithm, Ranking with Auxiliary Data (RAD).

3. EXPERIMENTAL EVALUATION

As a generic ranking algorithm, RAD can be applied to different ranking applications with different base learners. In this section, we apply RAD to Web search data with a popular base learner, regression tree, to demonstrate the properties and effectiveness of RAD.

We compare our algorithms with the following fourth approaches. The first one is the baseline approach using the target domain data only (called TD). The second one is another baseline approach simply using the source domain data (called SD) only. The third ones is an effective transfer learning algorithm based on Optimal Combination of source domain data and target domain data (called OC) [1, 3]. The fourth one is a state-of-the-art ranking approach using unlabeled data based on Self-Training (ST) [4].

We use Web search data from a commercial search engine. In the data, each query-document example is represented by a feature vector. Those query-document examples are originally labeled using a five-grade labeling scheme: {Bad, Fair, Good, Excellent, Perfect}. We then transform them into pairwise preference data. We select to use four domains corresponding to four different countries/languages, one for the target domain T1 and three for the source domains, S1, S2, and S3. The target domain T1 has 78,836 examples; each of source domains S1, S2, and S3 has about 20k examples. We randomly divide T1 into three subsets, about 10% for labeled data $\mathcal{T}$, about 80% for unlabeled data$\mathcal{U}$, about 10% hold on for test.

For the performance measure for ranking models, we select to use Discounted Cumulative Gain (DCG), which has been widely used to assess ranking quality in the context of search engines. Specifically, we use DCG-k, since users of a search engine are only interested in the top-k results of a query rather than a sorted order of the entire document collection. In this study, we select k as 5. For every experimental setting, 10 runs are repeated and in each run the target domain data set is randomly divided into the three subsets as mentioned before. The average DCG of 10 runs is reported for each experiment setting.

The ratio of weight coefficients α and β controls relative weight of unlabeled data w.r.t the labeled data in the target domain. Intuitively, this ration is important, since it is directly affect training of the ranking function h. On the other hand, the weight coefficients $\beta^{(j)}$ does not directly affect the training of h. In fact, in our experiments we observe that the result is not sensitive to $\beta^{(j)}$.

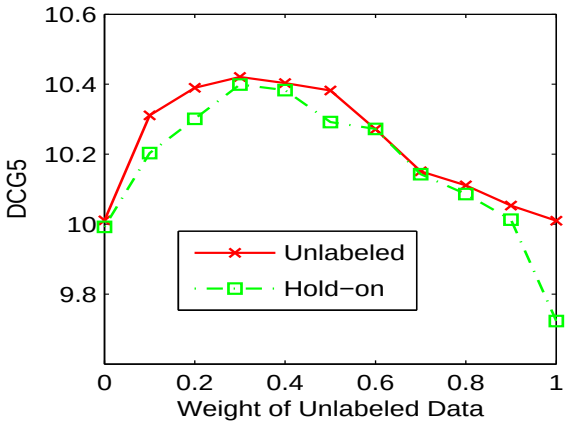

Figure 1: Effect of relative weight of unlabeled data against the total weight of labeled data and unlabeled data in the target domain.

Hence, we carry out experiments to show how relative weight of unlabeled data affects the performance of resulting h. We fix source domain data set as data set $S1$ and change the ratio of $\frac{\alpha}{\alpha+\beta}$, i.e, the relative weight of unlabeled data against all the data in the target domain as shown in the X axis in Figure 1. We test the learned h on both unlabeled data set and hold-out data set. Test on unlabeled data evaluate effectiveness of transduction learning, i.e., ranking examples we can observe before we start training process. This corresponds to real applications such that off-line ranking of very import query-documents. Test on hold-out data set evaluates the effectiveness of ranking on new examples. Those two types of two test results correspond to two curves in Figure 1.

Figure 1 shows that the relative weight of the unlabeled data is not monotonically related to the performance. The possible explanation for this is that unlabeled data have both useful information and noisy information; when weigh of unlabeled data reaches certain point, more noisy information will be transferred into h and hence leads to performance decrease. For example, at the extreme case that relative weigh of unlabeled is 1, i.e., the training data for h are just all the unlabeled data with pseudo-labels from the source domain function; hence inaccurate information from the source domain function will be transfered into h. We observe that when relative weight equals to 0.3, the best performance is archived. In all the following experiments, we use 0.3 this optimal relative weigh.

we also observe that performance of h on the unlabeled data is better than on the hold-out data. This verifies a part of our model assumption that the more similar distribution of unlabeled data is as the distribution of future new examples, the more useful information the ranking function h can obtain from unlabeled data. Test on unlabeled data simulates the idea case that the distribution of the unlabeled data is exactly the same as the distribution of the test data. Hence, test on unlabeled data should provide better (or at least equal in the case that unlabeled and hold-out data hap-

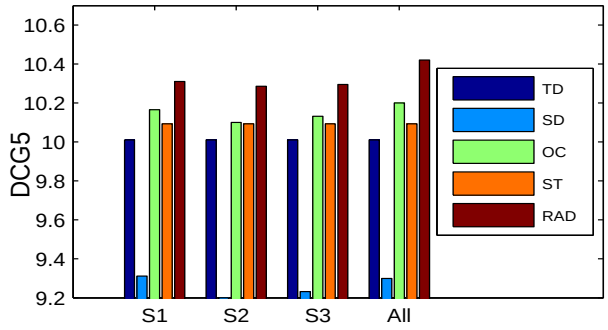

Figure 2: Test on unlabeled data: comparing the RAD algorithm with other algorithms on different source domains shows that the RAD algorithm performs best on different target domains.

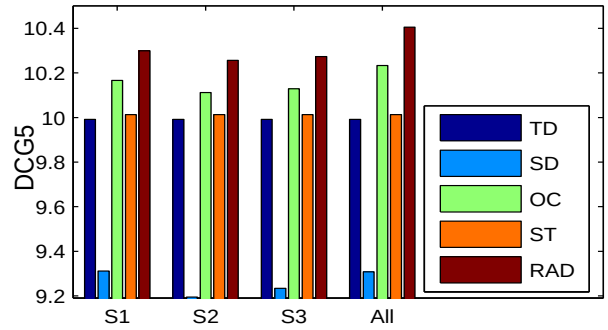

Figure 3: Test on hold-out data: comparing the RAD algorithm with other algorithms on different source domains shows that the RAD algorithm performs best on different target domains.

pen to have the same distribution) performance than the hold-out data.

Finally, we compare the RAD algorithms with other algorithms on different source domains. By using different source domains, we have different distances between the target domain and source domain. Based on the prior knowledge about the languages/countries associated with these domains, the difference between S1 and T1 is the smallest. In fact, they are from two countries with same languages; and S2 and S3 are different languages from T1.

From Figure 2 and From Figure 3, we observe the similar results except that for the algorithms, SF and RAD, involving unlabeled data, performance on hold-out data is worse.

For the both figures, we observe that the RAD algorithm provides best performance for three source domains and the combination of all three domains. For the source domain S1, which are similar to the target domain T1, the algorithms, SD, OC, and RAD, perform better than for the other two source domains, S2 and S3. This is consistent with the intuition that more close the source domain is, more it is.

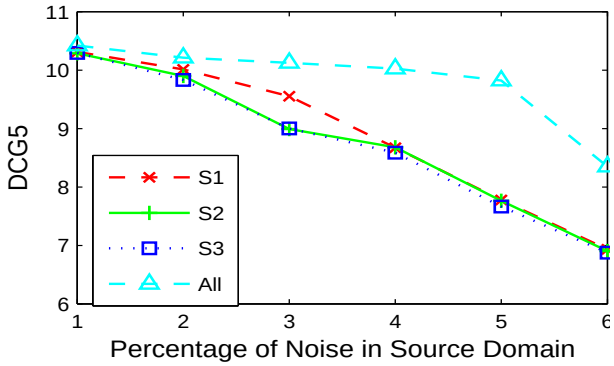

Figure 4: Effect of different levels of noise of the source domain data on DCG5 of the RAD algorithm shows that the multiple source domain data lead to more robust models.

We can observe that RAD performs better with the combination of three source domain data than with any single domain. The possible reasons are (1) the combination of three source domain data have a larger size and (2) the consistence among three source domains and the target domain provides more robust information than consistence between a single source domain and the target domain (hence, it leads to more robust ranking function h). We carry out the following experiments to verify the second point. We test the robustness of RAD algorithm under different source domains by intentionally adding noise into the source domain data. We add noise as follows. A preference pair is randomly selected, and its preference order is reversed. We change different percentages of the source domain data to create different levels of noise. Figure 4 shows that when the noise in the source domain data increases, the performance of the ranking model decrease; however, for multiple source domain data, the performance decreases much less, i.e., the ranking function trained with multiple source domain data is more robust.

4. REFERENCES

[1] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. *Advances in Neural Information Processing Systems*, 20, 2008.

[2] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[3] G. Schweikert, C. Widmer, B. Schölkopf, and G. Rätsch. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. In *NIPS*, pages 1433–1440, 2008.

[4] V. Truong, M. R. Amini, and P. Gallinari. A self-training method for learning to rank with unlabeled data. In *11th European Symposium on Artificial Neural Networks*, Bruges, Avril 2009.