Flexible Sample Selection Strategies for Transfer Learning in Ranking

Kevin Duh, Akinori Fujino

NTT Communication Science Laboratories
2-4 Hikaridai, Keihanna Science City
Kyoto 619-0237, JAPAN

Abstract

Ranking is a central component in information retrieval systems; as such, many machine learning methods for building rankers have been developed in recent years. An open problem is transfer learning, i.e. how labeled training data from one domain/market can be used to build rankers for another. We propose a flexible transfer learning strategy based on sample selection. source domain training samples are selected if the functional relationship between features and labels do not deviate much from that of the target domain. This is achieved through a novel application of recent advances from density ratio estimation. The approach is flexible, scalable, and modular. It allows many existing supervised rankers to be adapted to the transfer learning setting. Results on two datasets (Yahoo's Learning to Rank Challenge and Microsoft's LETOR data) show that the proposed method gives robust improvements.

Keywords:
Rank Algorithms, Transfer Learning, Sample Selection, Functional Change

Email addresses: kevin.duh@lab.ntt.co.jp (Kevin Duh), fujino.akinori@lab.ntt.co.jp (Akinori Fujino)

1. Introduction

The ranker is at the heart of many information retrieval systems. Its task is to order a list of documents such that the ones most relevant to the query appears first. Numerous machine learning techniques have been developed for building rankers. However, most work thus far focus on the traditional supervised learning setting, where identical distributions in training and test are assumed. An increasingly important problem is transfer learning, i.e. how labeled training data from one domain/market can be leveraged to improve ranking in another domain. For example, a search engine developer may desire to use its abundant data for its English-speaking market to bootstrap a ranker for its Chinese market.

This work proposes a transfer learning strategy for ranking. In particular, we focus on the following scenario, where two datasets both *with relevance judgment labels* are available:

- (A) Small dataset for target domain of interest (e.g. Chinese market)
- (B) Large dataset for another domain, which we term "source domain" (e.g. English market).

The goal of transfer learning is to build a ranker that performs well on the target domain using both datasets (A) and (B). We are deemed successful if this ranker outperforms a ranker trained only on dataset (A), on a test set coming from the same distribution as (A). This scenario is also termed

"supervised domain adaptation" in some literature since the target domain contains labels; here we use the general term "transfer learning."

Transfer learning is a challenging problem because the traditional machine learning theory which states that "minimizing (regularized) empirical risk leads to lower generalization error" [1] is no longer valid in the case when training and test distributions differ. Transfer learning is also a rewarding problem because solving it has not only theoretical importance, but also practical ramifications in terms of building better systems with lower cost.

We believe there are several criteria that ought to be satisfied for transfer learning in ranking: First, the proposed method ought to be able to incorporate many of the existing research in supervised ranking. Much effort has been devoted to developing better algorithms in the non-transfer setting, and it would be a waste to re-invent the wheel. Second, the proposed method should be computationally scalable so that large amounts of source domain data could be effectively exploited. Finally, ease of implementation is always a plus (but not a necessity), as it allows for faster development and experimentation cycles.

We propose a transfer learning strategy (based on sample selection) that satisfies all the above three criteria. Intuitively, our strategy first selects labeled source domain samples that are "related" to the target domain, then train a conventional ranker on the combined data. We define two ranked lists as "related" if the labeling process is similar (i.e. the definition of what is a relevant document is the same). We will show that this nicely models the so-called **"functional change assumption"** in transfer learning, and we compute this relatedness measure by employing state-of-the-art algorithms

from density ratio estimation [2].

In the following, Section 2 explains the intuition and algorithm of our proposed transfer learning strategy. Then, Sections 4 and 5 describes experimental results on the Yahoo Learning to Rank Challenge and the Microsoft LETOR datasets. Finally, we conclude with discussions about related work (Section 3) and future directions (Section 6).

2. Sample Selection for Ranking

2.1. Notation

In the machine learning approach to ranking, we are given datasets consisting of queries, document lists, and relevance judgment labels. Specifically, let q_i be a query and $\mathbf{X}_i$ be the corresponding list of documents retrieved by an initial search engine. We can think of $\mathbf{X}_i$ as a $K \times D$ matrix representing the D-dimensional feature vectors of the K query-document pairs. An annotator provides a relevance label to each query-document pair, forming the vector $\mathbf{y}_i \in \mathcal{R}^K$. The triple $(q_i, \mathbf{X}_i, \mathbf{y}_i)$ is referred to as a "sample" and indicates the desired ranking of a list of documents, given a particular query.

We have two datasets in transfer learning. The target domain training data consists of N^t samples $S^t = \{(q_i^t, \mathbf{X}_i^t, \mathbf{y}_i^t)\}_{i=1,\ldots,N^t}$. The source domain training data consists of N^s samples $S^s = \{(q_i^s, \mathbf{X}_i^s, \mathbf{y}_i^s)\}_{i=1,\ldots,N^s}$. Throughout the paper, we use the subscript i as index for the samples and the superscripts t/o to indicate target and source domain.

In the non-transfer setting, the dataset S^t is fed to a machine learning algorithm to generate a ranker, parameterized by the vector $\mathbf{w}^t$. We assume a *linear* ranker throughout this paper, so documents are ranked by order of

the dot product between $\mathbf{w}^t$ and query-document pair feature vector. In the transfer setting, the dataset S^s (or subsets thereof) is also incorporated. Our proposed transfer learning strategy amounts to selecting a subset of samples in S^s to be added to S^t, and training a ranker $\mathbf{w}^{t+s}$ based on the combined dataset. Generally, S^s is large while S^t is scarce (i.e. $N^s \gg N^t$).

2.2. Intuition

The issue with a small target dataset S^t is that the observed $\{q_i^t, \mathbf{X}_i^t\}_{i=1,\ldots,N^t}$ does not fully cover the whole range of queries and documents that may exist in the domain of interest. The task of machine learning, of course, is to extrapolate from finite samples, but the accuracy suffers if the dataset is simply too small.

The idea of our sample selection strategy is to *increase the coverage* by adding samples from source domain S^s. However, one needs to be very careful because the labels of an source domain sample $(q_i^s, \mathbf{X}_i^s, \mathbf{y}_i^s)$ may lead to a ranker $\mathbf{w}^{t+s}$ that is very different from the *true* target domain function (initially approximated by $\mathbf{w}^t$). source domain samples may be labeled differently because the definition of relevance may vary across domains. For example, for a query submitted to a multilingual collection, what is relevant depends on whether the annotator prefers Chinese or English documents. This is known as the "functional change" assumption in transfer learning [3].

Our strategy only adds source domain samples that are "related" or "close" to the target domain. The key to defining "relatedness" is to note that a *sample-specific* ranker $\mathbf{w}_i$ (i.e. a ranker trained on just a single sample $(q_i, \mathbf{X}_i, \mathbf{y}_i)$) can characterize the functional relationship between $\mathbf{X}_i$ and $\mathbf{y}_i$. We assume that two samples are more related if their $\mathbf{w}_i$ are similar. Thus

our sample selection strategy works as follows: First, train a linear ranker independently for each sample. The trained rankers/weights of the target domain $\mathbf{w}_i^t$ forms a probability density distribution in $\mathcal{R}^D$ space. source domain sample rankers $\mathbf{w}_i^s$ appearing near dense regions of $\mathbf{w}_i^t$ are more likely to be selected. *These selected samples are those that may increase the coverage of the training set but do not significantly change the functional relationship between features and labels.*

Figure 1 illustrates the intuition using Microsoft's LETOR dataset. The target domain is a topic distillation task, while source domain is a homepage finding task.[1] The plot represents the distribution of weights given by sample-specific rankers for two features. We observe that while there are some overlap between the target and source domain distributions, there are some source domain samples that are outliers with respect to the target distribution. For example, some source domain samples place large negative weights on the PageRank feature compared to what is usually given by target domain samples. These source domain outliers are the ones that will *not* be selected by our sample selection strategy.

In general, in transfer learning there are two main assumptions that *explicitly clarify* how target and source domains differ. Here we address the "functional change" assumption, which says that the *labeling process* changes

[1]Topic distillation aims to find "overview" webpages, whereas homepage finding aims to retrieve specific webpages. Consider the query "Data mining conference": A good result for topic distillation may be a resourceful page that lists all data mining conferences of the year; but a good answer for homepage finding might be a particular conference website, e.g. "International Conference on Data Mining."

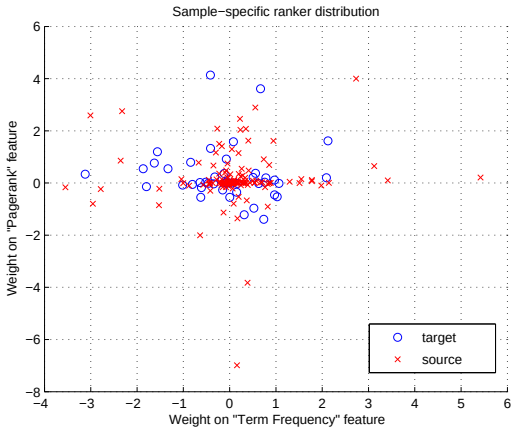

Figure 1: Distribution of sample-specific rankers/weights. Note that some source domain samples are outliers with respect to the target distribution.

between source and target, i.e. given the same $\{q, \mathbf{X}\}$, the labels $\mathbf{y}$ will vary among domains. This is natural when the definition of relevance changes, such as when different search tasks are run (topic distillation vs. homepage finding) or when different languages are preferred (English vs. Chinese). Another popular assumption, "covariate shift", states that the *input* distribution changes: e.g., the queries that are popular for the English market may not necessarily also be popular for the Chinese market. Of course, both assumptions may be at play in practice, but we are not aware of any method that can simultaneously tackles both.

2.3. Algorithm

The intuition in Section 2.2 and Fig 1 is implemented by the following pseudocode (see Algorithm 1). In the first phase (lines 1-6), rankers are trained for each sample using a generic supervised learning algorithm,

RANK-LEARN(). This algorithm could be, e.g. RankSVM, or any other supervised algorithm that generates linear rankers. This phase is scalable/parallelizable even as the number of source domain sample increases because it is done independently for each sample; further, training on single queries is extremely fast.

The second phase (Algorithm 1, line 7) aims to quantify the "relatedness" between the collected linear weights for target and source domain samples. A natural way to do this is to use the so-called density ratio estimation methods [2], which quantifies differences between samples from different distributions: for each source domain weight $\mathbf{w}_i^s$, we estimate the ratio:

$$r_i^s = \frac{p^t(\mathbf{w}_i^s)}{p^s(\mathbf{w}_i^s)} \tag{1}$$

where $p^t(\cdot)$ is the probability density function for target domain weights, and $p^s(\cdot)$ is the probability density function for source domain weights. In the example of Figure 1, we can thus imagine two distributions in $\mathcal{R}^2$. For the regions where PageRank < -2, $p^t(\cdot)$ is likely to be smaller than $p^s(\cdot)$, leading to a small ratio r_i^s. Samples with small r_i^s are less related to the target domain.

The final phase (line 8 to 22 of Algorithm 1) finds the best subset of source domain samples to be added to the target domain data. This is done by using a threshold α and selecting sample i if $r_i^s \geq \alpha$ (lines 10-16). An overall ranker is re-trained on the combination of source subset and target data (line 16). The best value of α is searched for on the development set

(line 17-20).[2]

In practice, the ratio r_i^s is not computed by estimating $p^s(\cdot)$ and $p^t(\cdot)$ explicitly since density estimation is a difficult problem. Instead, we follow a state-of-the-art method called Kullback-Liebler Importance Estimation Procedure (KLIEP) that directly computes r_i^s [2]. First, we define a function $r^s(\mathbf{w})$, where $r_i^s = r^s(\mathbf{w}_i)$, and parameterize it by a linear combination of kernels $r^s(\mathbf{w}) = \sum_{j=1}^{N^t} \gamma_j k(\mathbf{w}, \mathbf{w}_j^t)$. Then we minimize the KL-divergence between the target distribution $p^t(\mathbf{w})$ and a *weighted* source domain distribution $r^s(\mathbf{w}) \cdot p^s(\mathbf{w})$:

$$
\begin{aligned}
\arg\min_{r^s(\mathbf{w})} KL(p^t(\mathbf{w}) \| r^s(\mathbf{w}) p^s(\mathbf{w})) &= \arg\min_{r^s(\mathbf{w})} \int p^t(\mathbf{w}) \log \frac{p^t(\mathbf{w})}{r^s(\mathbf{w}) \cdot p^s(\mathbf{w})} d\mathbf{w} \\
&= \arg\min_{r^s(\mathbf{w})} \int p^t(\mathbf{w}) \log \frac{p^t(\mathbf{w})}{p^s(\mathbf{w})} d\mathbf{w} - \int p^t(\mathbf{w}) \log r^s(\mathbf{w}) d\mathbf{w} \\
&= \arg\min_{r^s(\mathbf{w})} - \int p^t(\mathbf{w}) \log r^s(\mathbf{w}) d\mathbf{w} \\
&\approx \arg\min_{r^s(\mathbf{w})} \frac{-1}{N^t} \sum_{i=1}^{N^t} \log r^s(\mathbf{w}) \\
&= \arg\min_{\gamma_j} \frac{-1}{N^t} \sum_{i=1}^{N^t} \log \sum_{j=1}^{N^t} \gamma_j k(\mathbf{w}, \mathbf{w}_j^t)
\end{aligned}
$$

This formulation in terms of KL-divergence allows us to directly compute r_i^s without explicitly calculating $p^t(\cdot)$ and $p^s(\cdot)$ in Eq. 1. We use Gaussian kernels $k(\mathbf{w}, \mathbf{w}_j^t) = \exp(-||\mathbf{w} - \mathbf{w}_j^t||/2\sigma)$ situated on the target weights $\mathbf{w}_j^t$. The above objective can be efficiently optimized by a convex

[2]In the pseudocode, all values of $\alpha = [0, max(r)]$ are searched. In practice, we found that searching for a few values in the interval is sufficient and speeds up training.

program, with additional constraints $\gamma_j \geq 0$ (so ratios are non-negative) and $1 = \int r^s(\mathbf{w}) p^s(\mathbf{w}) d\mathbf{w} \approx \frac{1}{N^s} \sum_{i=1}^{N^s} \log \sum_{j=1}^{N^t} \gamma_j k(\mathbf{w}, \mathbf{w}_j^t)$ (so the distribution properly sums to one). Finally, the optimal values γ_j are used to compute the ratio r_i^s for each source domain sample.

In summary, density ratio estimation provides a way to compute the relatedness of samples, where relatedness is based on whether the sample-specific ranker occurs in a region where $\frac{p^t(\mathbf{w})}{p^s(\mathbf{w})}$ is high. Samples with high ratios are good candidates to be added to the training set, because they increase coverage with less risk of changing the functional relationship.

The reader may wonder why we chose samples with high ratios $\frac{p^t(\mathbf{w})}{p^s(\mathbf{w})}$, rather than simply high density $p^t(\mathbf{w})$. The reason is that our sample selection strategy performs a simple linear search α for the threshold, so it is important to discount regions of high $p^s(\mathbf{w})$ in order to avoid over-sampling, which introduces bias. Another view is to consider the case when source and target distributions are actually equal; then it makes sense to look at the ratio, which is close to 1 for all samples.

3. Related Work

Most work in transfer learning for ranking are *learner-specific*, i.e. based on modifying a particular supervised ranking method: For example, Geng et. al. [11] modifies the regularization term of a RankSVM to ensure that the learned target ranker is close to the source domain ranker. For boosted tree rankers, Gao [12] and Chen [13] use target domain data to re-tune a tree originally trained on source domain data; Chapelle et. al. [14] extend boosted tree adaptation to a multitask learning setting, where multiple source domain

Algorithm 1 Transfer Learning in Ranking by Sample Selection

Input: Target training data: $S^t = \{q_i^t, \mathbf{X}_i^t, \mathbf{y}_i^t\}_{i=1,...,N^t}$

Input: Target development data: $S^v = \{q_i^v, \mathbf{X}^v, \mathbf{y}^v\}_{i=1,...,N^v}$

Input: source domain training data: $S^s = \{q_i^s, \mathbf{X}_i^s, \mathbf{y}_i^s\}_{i=1,...,N^s}$

Output: Ranker $\mathbf{w}^{t+s}$

for $i = 1, ..., N^t$ **do**
  $\mathbf{w}_i^t$ = RANK-LEARN($(q_i^t, \mathbf{X}_i^t, \mathbf{y}_i^t)$) # compute query-specific rankers
end for
for $i = 1, ..., N^s$ **do**
  $\mathbf{w}_i^s$ = RANK-LEARN($(q_i^s, \mathbf{X}_i^s, \mathbf{y}_i^s)$) # compute query-specific rankers
end for
$\mathbf{r}^s$ = DENSITY-RATIO($\{\mathbf{w}_i^t\}_{i=1,..,N^t}, \{\mathbf{w}_i^s\}_{i=1,..,N^s}$) # find weights
$\beta = 0$
for α = 0,..,max($\mathbf{r}^s$) **do**
  $S^{t+s} \leftarrow \{S^t\}$
  for $i = 1, ..., N^s$ **do**
    if $\mathbf{r}_i^s \geq \alpha$ **then**
      $S^{t+s} \leftarrow \{S^{t+s}, (q_i^s, \mathbf{X}_i^s, \mathbf{y}_i^s)\}$ # select source subset
    end if
  end for
  $\hat{\mathbf{w}}$ = RANK-LEARN(S^{t+s}) # train overall ranker
  $\hat{\beta}$ = ACCURACY($\hat{\mathbf{w}}, S^v$)
  if $\hat{\beta} \geq \beta$ **then**
    $\beta = \hat{\beta}$; $\mathbf{w}^{t+s} = \hat{\mathbf{w}}$ # choose overall ranker based on development set
  end if
end for
Return $\mathbf{w}^{t+s}$

datasets are jointly learned. These methods are promising, but it forces the designer to be fixed to a particular algorithmic framework. The way these methods exploit source domain data is to use it to improve the estimation of model parameters (e.g. by regularization, or by fixing a tree structure).

We are not aware of much *learner-independent* work like ours (where various rankers could be plugged-in to a general transfer learning framework), besides [10] which adopts a feature duplication approach [9]. The idea is to duplicate the features so that every feature has a target, source domain, and shared version. For example, if there are 700 features originally, we would now get 2100 features: for the target data, the first 1400 feature are active (and feature id f is the same value as feature id $f + 700$, for $1 \geq f \geq 700$). For source domain data, the final 1400 features are active (and feature id $f + 700$ equals feature id $f + 1400$, for $1 \geq f \geq 700$). By doing so, a learning algorithm (with suitable regularization) will focus weights on shared features when possible, and put weights on the other two when domain differences dominate. In general, it should be possible to embed the learner-specific methods into the learner-independent approaches. Learner-independent approaches usually exploit source domain data by focusing on the features (e.g. [10]) or the samples (this work).

There are some learner-independent work in settings slightly different from transfer learning here: Wang et. al. [15] framed a "heterogeneous" ranking problem where different domains contain fundamentally different types of objects (e.g. documents vs. citation network). The focus is on finding a latent space mapping so that distinct feature representations could be shared. In semi-supervised ranking (where some documents have no labels),

[16] also proposed to use density ratio estimation, albeit with very different assumptions. There the goal is to estimate density ratios between labeled and unlabeled features $\mathbf{X}$, whereas here we estimate in the function space of $\mathbf{w}$.

The local learning approach [17, 18] is similar to ours in the sense that each query/document-list $\{q_i, \mathbf{X}_i, \mathbf{y}_i\}$ is treated as an independent unit. For example, Geng et. al. [17] selects training samples based on query (q_i) similarity and Banerjee et. al. [18] does so based on document ($\mathbf{X}_i$) similarity; the selected samples are used for constructing rankers tuned to the test query. On the other hand, our approach selects samples based on the functional relationship, $\mathbf{w}_i$, which is derived from $\mathbf{X}_i$ and $\mathbf{y}_i$.

Density ratio estimation is a relatively new area within machine learning. Besides the KL-based method use here, other methods include mean matching [19], discriminative modeling [20], and least squares fitting [21]. These techniques have been applied to covariate shift adaptation, feature extraction [23], and outlier detection [22]. However, to the best of our knowledge, the current paper is the first work to use density ratio estimation to explicitly model the functional change assumption in transfer learning.

Most closest to our work is perhaps the classification work by Jiang and Zhai [24]. They also focused on the functional change assumption: the approach first trains a classifier on target data, then selects source domain samples that are not misclassified by it. This approach can be extended to ranking, as we will do the experiments (Section 5), by training a ranker on all of the target data and selecting source domain queries that have high NDCG under this ranker. Another closely related work is by Long et. al. [25], who

reweights the source domain samples by comparing the outputs of source domain and target domain regressors/classifiers. If the outputs are similar, then the sample has deemed more useful and given more weight. A very nice contribution of [25] is a formal risk minimization objective that can be optimized by a general boosting framework. Our method differs from these two works in that (1) we train multiple local rankers as opposed to one or two source/target rankers, and (2) we directly compare distributions in function space, as opposed to the label's output space. The first aspect allows us to look at a distributions rather than points, while the second aspect allows us to compare functions rather than outputs. Thus, we call our method "Sample selection by Function Distribution" and the methods of [24, 25] "Sample selection by Label Relation". Both address the transfer learning problem of functional change. We will not argue that ours is better; it is just two different approaches.

4. Experiment Setup

4.1. Datasets

We performed experiments on two datasets. The data from the Yahoo! Learning to Rank Challenge (Track 2)[4] consists of data for a real-life search engine in two different markets. This data was available as part of a open challenge in Spring 2010; we have divided the original target training data randomly into five folds in order to perform cross-validation in our experiments. Target training data consists of roughly 1000 queries, and source domain data is twenty-times larger. In order to evaluate transfer learning under various resource settings, we also perform data ablation experiments

where target training data is artificially reduced while source domain data is held fixed.

Our second data comes from the publicly-available Microsoft LETOR distribution v3.0 [5]. To create a transfer learning scenario, we take the topic distillation task (2004td) as the target domain; the source domain data consists of the named paged finding (2004np) and homepage finding tasks (2004hp). All data come from the TREC2004 Web track, which represents retrieving webpages on the .GOV collection. The domains are different in the sense that topic distillation finds webpages that are good information hubs on a certain topic, whereas the homepage and named page tasks require finding particular URLs (navigational queries). We mix both homepage and named page finding data to test our method on *heterogeneous* source domain data. source domain data is 3.3 times the size of target data.

Table 1 presents the data statistics. For the Yahoo data, there are five levels of relevance judgments (ranging from "very relevant" to "irrelevant") and 700 features. Some features are defined only for the target domain (596 features) or only for source domain (519 features); the feature value is set to zero if it is not defined to keep the feature vector fixed at 700. For the LETOR data, there are two levels of relevance and 64 features, including term frequency, BM25, LMIR, and PageRank. Note that LETOR has fewer target queries (45) compared to the Yahoo dataset (1012), but more documents per query.

4.2. Evaluation Metric

We use Normalized Discounted Cumulative Gain (NDCG) [6], a popular evaluation metric for information retrieval:

YAHOO Dataset	#Queries	#Documents	#Features
Target Training data	1012	27,852	700
Target Development data	127	3,638	700
Target Test data	127	3,325	700
Source Training data	19,944	473,134	700
LETOR Dataset	#Queries	#Documents	#Features
Target Training data	45	44,487	64
Target Development data	15	14,829	64
Target Test data	15	14,829	64
Source Training data	150	148,243	64

Table 1: Dataset statistics

$$NDCG = \frac{DCG}{IdealDCG} \qquad DCG = \sum_{m=1}^{M} \frac{2^{y_m} - 1}{\log_2(1+m)} \tag{2}$$

where y_m is the label value for a document ranked at position m. DCG is normalized by IdealDCG, which is the value for perfect ranking, so that NDCG ranges between $[0, 1]$. Intuitively, NDCG will be large if the most relevant documents (with high y_m) are ranked at the beginning. Here we average up to the first M=5, 10, and 15 positions (NDCG@5, NDCG@10, NDCG@15).

We use the python/perl scripts associated with each dataset to compute the evaluation metrics. The Yahoo script additionally reports Expected Reciprocal Rank (ERR) [7] while the LETOR script reports Mean Averaged Precision (MAP). ERR measures the expected reciprocal document position

when an user becomes satisfied, assuming a cascaded model (linear scan of documents). MAP calculates the precision of relevant documents averaged across various positions. We treat NDCG@10 as the main evaluation criterion.

5. Results

5.1. Main Results

For all experiments, we use RankSVM [8] as the base ranker (RANKLEARN()). The parameters are the threshold α and SVM cost tradeoff c, both of which are tuned on the development set. For the threshold α, we tuned for values that correspond to the $50, 60, 70, 80, 90$ percentiles of $\mathbf{r}^s$. For the SVM cost tradeoff c, we choose among $\{0.01, 0.1, 1, 10, 100\}$. All results show averaged 5-fold results on the test set.

We compare the following domain adaptation methods:

1. **SampleSelect (Function)**: Sample selection by Function Distribution. This is the **proposed approach** which selects source samples by comparing densities in function space..
2. **SampleSelect (Label)**: Sample selection by Label Relation. This method extends the classification algorithm of Jiang and Zhai [24] to ranking. A ranker trained on all of target data is used to rank source data. For each source query, we then compute the NDCG/MAP scores. Source samples with high scores are selected and folded into the training data. The amount of samples to select is determined by cross-validation, similar to the proposed method.

3. **Feature duplication**: This is a learner-independent transfer learning method that has achieved state-of-the-art results [9, 10].

For reference, we also have the following simple baselines:

1. **Target-only**: Training on only target data (non-transfer)
2. **Source-only**: Training on only source domain (mismatch condition)
3. **Combined Data**: Training on combined target and source data
4. **Weighted Combined**: Training on a weighted combination of target and source data. In particular, target data is weighted to be 1, 1.5, 2, 2.5 or 3 times the number of source queries, and the best result is chosen by cross-validation. (E.g. for Yahoo data which has about 1k target vs 20k source queries, we would multiply the target data by multiples of 20, 30, 40, 50, 60, respectively.)

The results for Yahoo dataset is shown in Table 2. We observe that the **Target-only** baseline gives .729 NDCG@10. Naively adding all source domain data (**Combined**) causes it to drop to .724 NDCG@10, while a tuned **Weighted Combined** improves it to .732 NDCG@10. On the other hand, the proposed **SampleSelect (Function)** strategy improves NDCG@10 to .735, implying that smart selection of data is important. This is a statistically significant improvement relative to **Weighted Combined**, according to the t-test, with $p < 0.05$.

The proposed method also outperforms the competing domain adaptation methods **SampleSelect (Label)** and **Feature Duplication**. It appears that sample selection by function distribution is better than by label relation, which is respectively better than feature duplication. The other NDCG positions and ERR metric shows similar trends.

System	NDCG@5	NDCG@10	NDCG@15	ERR
SampleSelect (Function)	**0.694**	**0.735**	**0.772**	**0.439**
SampleSelect (Label)	0.690	0.731	0.769	0.438
Feature Duplication	0.690	0.731	0.731	0.437
Target-only	0.690	0.729	0.768	0.437
Source-only	0.655	0.701	0.741	0.406
Combined Data	0.686	0.724	0.759	0.432
Weighted Combined	0.693	0.732	0.772	0.438

Table 2: Cross-validation Results (Yahoo data)

The results for the LETOR dataset are shown in Table 3. Contrary to the Yahoo result, adding all the source domain data (**Combined Data**) did not degrade the **Target-only** results (both are at .310NDCG@10). **SampleSelect (Function)** gave some improvements (.314 NDCG@10), but in general it did not outperform the baselines by statistically significant margins, contrary to case of the Yahoo dataset result. Comparison with other domain adaptation results are inconclusive, with the proposed method winning in NDCG@10 and NDCG@15 but not in other metrics. We conjecture that the source domain data in this dataset may be too small for us to observe noticeable effects. The fact that adding all source domain data without any smart selection did not degrade results lead us to believe that (1) the domains may be quite similar, and (2) the target training data is sufficiently small so that any extra data may be beneficial.

Summary: Our proposed **SampleSelect (Function)** consistently im-

System	NDCG@5	NDCG@10	NDCG@15	MAP
SampleSelect (Function)	0.344	**0.314**	**0.315**	0.226
SampleSelect (Label)	**0.346**	0.310	0.312	0.226
Feature Duplication	**0.346**	0.312	0.312	0.226
Target-only	0.330	0.310	0.312	0.226
Source-only	0.327	0.306	0.302	0.213
Combined Data	0.345	0.310	0.309	**0.227**
Weighted Combined	0.335	**0.314**	0.313	0.224

Table 3: Cross-validation Results (LETOR data)

proved results or tied performance with the baseline **Target-only** or **Weighted Combined** systems, according to various metrics on the two datasets. For the larger Yahoo dataset, the proposal additionally outperforms other state-of-the-art transfer learning methods **SampleSelect (Label)** and **Feature Duplication**. It is therefore a very robust approach to transfer learning.

5.2. Detailed Analysis

We performed additional experiments to analyze our method in more detail, focusing on the Yahoo dataset:

5.2.1. How does the method work when amount of target training data varies?

Figure 2 shows the results for data ablation experiments. We artificially reduced the size of the target training data by 0.2, 0.4, and 0.8 fraction of samples. The fractions 0.2, 0.4, and 0.8 corresponds to roughly 110-140, 260-290, and 550-780 queries, respectively (the numbers vary by cross-validation

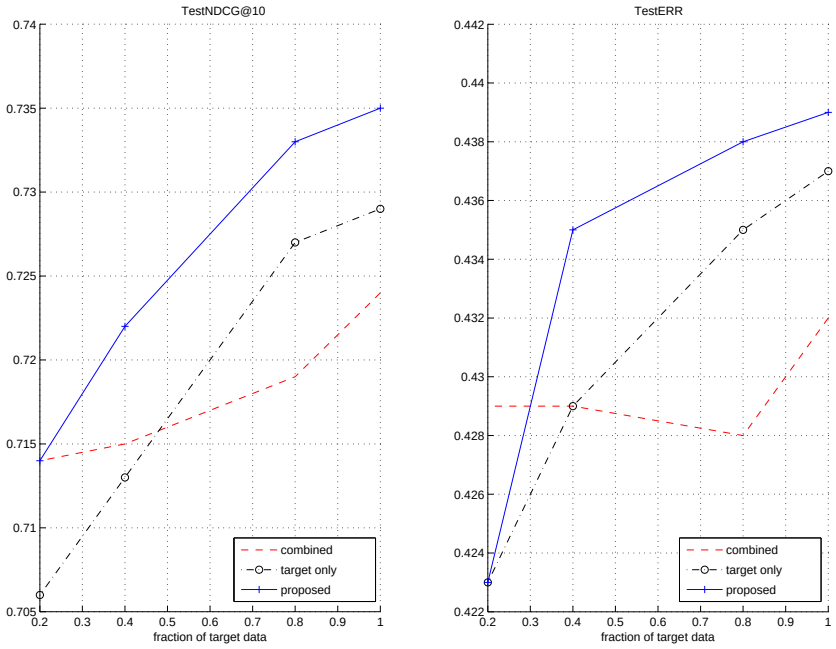

Figure 2: Data ablation results on Yahoo data: Proposed sample selection outperforms others on 0.4, 0.8, and 1 fraction of data.

fold). We are interested to see whether **sample selection** works well for a variety of scenarios regarding data resources, in order to find the best operating region of this method. The figure shows that **sample selection** gave robust improvements over almost all conditions. For 0.4 and 0.8 fraction of data, it outperformed both **combined** and **target-only**, just like it did for the full data case; the results are tied for the 0.2 condition. We conclude that sample selection is competitive for a variety of target training data amounts, and especially good for larger training datasets.

5.2.2. How important is the density ratio estimation step?

Recall that our method performs density ratio estimation in the space of model parameters $\mathbf{w}$. How important is density ratio estimation, per se? One may imagine other ways to sample source domain data, not using density ratio estimation. Here we answer this question by comparing with a straightforward approach: First, we obtain an average weight vector for the target domain by taking the mean of the trained vectors: $\mathbf{w}^t_{avg} = \sum_{i=1}^{N^t} \mathbf{w}^t_i$. Then, we simply choose source domain samples that have $\mathbf{w}^s_i$ close to $\mathbf{w}^t_{avg}$ (i.e. small $||\mathbf{w}^t_{avg} - \mathbf{w}^s_i||$). This straightforward scheme represents another way to select source domain samples in the weight space, but not using the machinery of density ratio estimation. Our goal is to see if this also gives good results, or if density ratio estimation is important.

Figure 3 shows the performance comparison, varied over different thresholds α (which corresponds different percentage of source domain samples selected). We see that for all cases, sample selection by density ratio estimation outperforms the straightforward method based on distance to the average vector. We thus conclude that taking into account the distribution of in-domain weight vectors is important. Looking at the distribution, rather than a mean statistic, allows us to better identify outliers in the source domain data.

5.2.3. What happens when the number of documents per query is reduced?

Our method trains query-specific rankers in order to characterize functional change, and this step relies on obtaining robust rankers on a single query. Thus, a concern is whether small numbers of documents per query will degrade the estimation of query-specific rankers. To test this, we artifi-

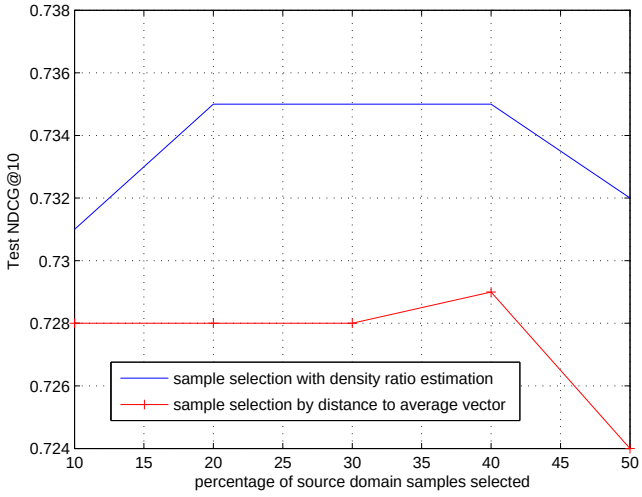

Figure 3: Comparison between density ratio estimation and simple distance to average approach for sample selection.

cially reduced the number of documents in the Yahoo dataset, in particular by deleting the bottom half of each query's document list. We performed density ratio estimation on rankers trained on these smaller datasets and compared the results with the original dataset. Naturally, the trained weight vectors will be different. But what is important is whether the ordering of density ratio values changes significantly, since this affects the subset of samples selected by our method.

Figure 4 shows how much of the selected subset of the reduced dataset overlaps with that of the original. This is calculated by sorting the source samples according to density-ratio values, and seeing how much intersection there is at each threshold. We can see that the overlap is quite high (e.g. when 50% of the source samples are selected, 90% of those selected are also

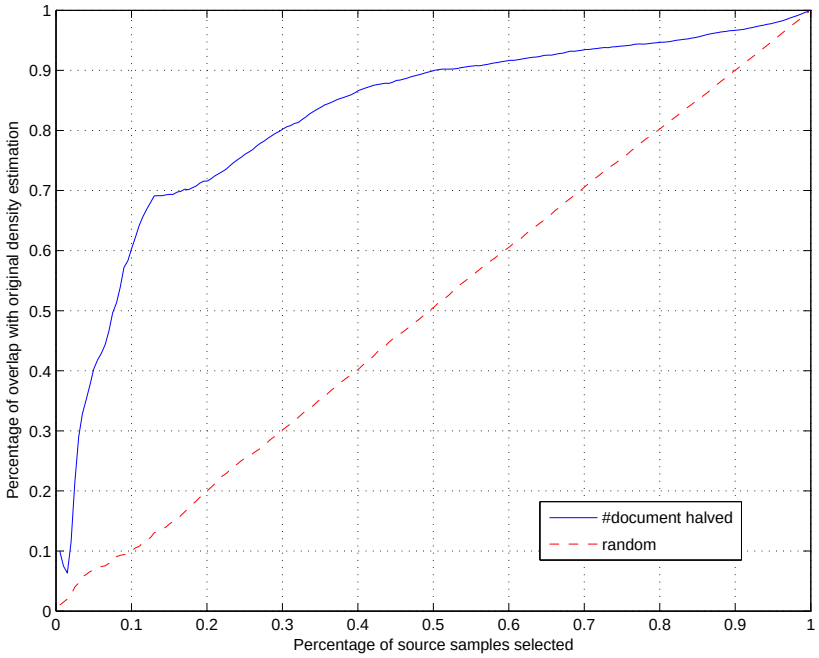

Figure 4: Measure of subset overlap between sample selection using density ratios trained by original vs. half of document/query training data (blue line). The red dashed line shows the overlap between the original selection and random selection.

present in the original subset). Thus it appears that our method is relatively robust to changes in number of documents per query. The subset using the density ratio values from reduced dataset achieved a test MAP of 0.438 and MAP of 0.734, which is relatively close to the original **Sample Select (Function)** result in Table 2.

6. Discussions and Conclusions

6.1. Summary

The contributions of this work are two-fold:

1. A general strategy to transfer learning for ranking based on sample selection: This is *scalable*, *flexible*, and *modular*, allowing the plug-in of many existing supervised ranking methods. The method achieves competitive or better results compared to state-of-the-art baselines.
2. A specific method based on density ratio estimation on sample-specific rankers $\mathbf{w}_i$. This method naturally captures the assumption of "functional change" in transfer learning, which has not been previously examined in the context of ranking.

We emphasize that our approach is learner-independent: sample selection can be used in conjunction with many of the existing ranking algorithms in the literature. Furthermore, the fact that we select samples in the model space $\mathbf{w}$, as opposed to the feature space, is novel and sets it apart from many previous approaches to transfer learning (for both classification and ranking). This allows us to capture the "functional change" assumption and incorporate *labeled information* in the transfer learning process.

6.2. Limitations and Extensions

One limitation here is the constraint on *linear* rankers. While linear rankers (e.g. [8, 26, 27, 28]) are popular in practice due to their strong performance and fast computation, we may sometimes desire non-linear rankers for their more expressive power. We require linearity because we use a Gaussian

kernel $k(\mathbf{w}_i, \mathbf{w}_j) = \exp(-||\mathbf{w}_i - \mathbf{w}_j||/2\sigma)$ to measure the similarity between rankers.

It is possible to extend this work to nonlinear rankers, such as trees [12], neural networks [29], and RankBoost [30], if we can define kernels that accurately capture the similarity between rankers. The criteria for such a kernel is: $\text{PRED}(w_i, S) \approx \text{PRED}(w_j, S) \Rightarrow k(\mathbf{w}_i, \mathbf{w}_j) \rightarrow 0$; i.e. rankers that are close in kernel space are those that give similar predictions on the same data. Concretely, we can let two rankers predict on the same set of queries, and use the L_2 distance between these prediction vectors to measure similarity.

Finally, one may also consider different methods for density ratio estimation. We experimented with the KL-based method here, but other methods such as mean matching [19], discriminative logistic regression modeling [20], and least squares fitting [21] may also perform well. An interesting avenue of future research is to better understand what kind of density ratio estimation algorithm is most suitable with this general ranker adaptation method.

References

[1] V. Vapnik, Statistical Learning Theory, Springer, 1998.

[2] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, M. Kawanbe, Direct importance estimation for covariate shift adaptation, Annals of the Institute of Statistical Mathematics 60 (4).

[3] J. Jiang, A Literature Survey on Domain

Adaptation of Statistical Classifiers, URL: http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey/, 2008.

[4] O. Chapelle, Y. Chang, Yahoo! Learning to Rank Challenge, URL: http://learningtorankchallenge.yahoo.com, 2010.

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

[6] K. Jarvelin, J. Kekalainen, IR evaluation methods for retrieving highly relevant documents, in: SIGIR, 2000.

[7] O. Chapelle, D. Metzler, Y. Zhang, P. Grinspan, Expected Reciprocal Rank for graded relevance, in: CIKM, 2009.

[8] T. Joachims, Training linear SVMs in Linear time, in: KDD, 2006.

[9] H. Daume, Frustrating easy domain adaptation, in: ACL, 2007.

[10] D. Chen, J. Yan, G. Wang, Y. Xiong, W. Fan, Z. Chen, TransRank: A Novel Algorithm for Transfer of Rank Learning, in: IEEE International Conference on Data Mining (ICDM), Workshop on Domain Driven Data Mining, 2008.

[11] B. Geng, L. Yang, C. Xu, X.-S. Hua, Ranking model adaptation for domain-specific search, in: CIKM, 2009.

[12] J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, H. Zhou, Model adaptation via model interpolation and boosting for web search ranking, in: ACL, 2009.

[13] K. Chen, R. Lu, C. Wong, G. Sun, L. Heck, B. Tseng, Trada: tree based ranking function adaptation, in: CIKM, 2008.

[14] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Wienberger, Y. Zhang, B. Tseng, Multi-task learning for boosting with application to web search ranking, in: KDD, 2010.

[15] B. Wang, J. Tang, W. Fan, S. Chen, Z. Yang, Y. Liu, Heterogenous cross domain ranking in latent spaces, in: CIKM, 2009.

[16] K. Duh, K. Kirchhoff, Semi-supervised ranking for document retrieval, Computer Speech and Language DOI: http://dx.doi.org/10.1016/j.csl.2010.05.002.

[17] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, H.-Y. Shum, Query dependent ranking using K-nearest neighbor, in: SIGIR, 2008.

[18] S. Banerjee, A. Dubey, J. Machchhar, S. Chakrabarti, Efficient and accurate local learning for ranking, in: SIGIR Workshop on Learning to Rank for Information Retrieval (LR4IR), 2009.

[19] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, B. Schölkopf., Correcting sample selection bias by unlabeled data., in: NIPS, 2007.

[20] S. Bickel, M. Brückner, T. Scheffer, Discriminative learning for differing training and test distributions, in: ICML, 2007.

[21] T. Kanamori, S. Hido, M. Sugiyama, A least-squares approach to direct importance estimation, in: Journal of Machine Learning Research, 2009.

[22] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, T. Kanamori, Statistical outlier detection using direct density ratio estimation, in: Knowledge and Information Systems, 2010.

[23] T. Suzuki, M. Sugiyama, Sufficient dimension reduction via squared-loss mutual information estimation, in: AISTATS, 2010.

[24] J. Jiang, C. Zhai, Instance weighting for domain adaptation in NLP, in: ACL, 2007.

[25] B. Long, S. Lamkhede, S. Vadrevu, Y. Zhang, B. Tseng, A risk minimization framework for domain adaptation, in: CIKM, 2009.

[26] J. Xu, H. Li, AdaRank: A boosting algorithm for information retrieval, in: SIGIR, 2007.

[27] D. Cossock, T. Zhang, Subset ranking using regression, in: COLT, 2006.

[28] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to rank: from pairwise to listwise approach, in: SIGIR, 2007.

[29] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: ICML, 2005.

[30] Y. Freund, R. Iyer, R. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4.