Query Weighting for Ranking Model Adaptation

Peng Cai[1], **Wei Gao**[2], **Aoying Zhou**[1], **and Kam-Fai Wong**[2,3]
[1]East China Normal University, Shanghai, China
pengcai2010@gmail.com, ayzhou@sei.ecnu.edu.cn
[2]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{wgao, kfwong}@se.cuhk.edu.hk
[3]Key Laboratory of High Confidence Software Technologies, Ministry of Education, China

Abstract

We propose to directly measure the importance of queries in the source domain to the target domain where no rank labels of documents are available, which is referred to as query weighting. Query weighting is a key step in ranking model adaptation. As the learning object of ranking algorithms is divided by query instances, we argue that it's more reasonable to conduct importance weighting at query level than document level. We present two query weighting schemes. The first compresses the query into a *query feature vector*, which aggregates all document instances in the same query, and then conducts query weighting based on the query feature vector. This method can efficiently estimate query importance by compressing query data, but the potential risk is information loss resulted from the compression. The second measures the similarity between the source query and each target query, and then combines these *fine-grained* similarity values for its importance estimation. Adaptation experiments on LETOR3.0 data set demonstrate that query weighting significantly outperforms document instance weighting methods.

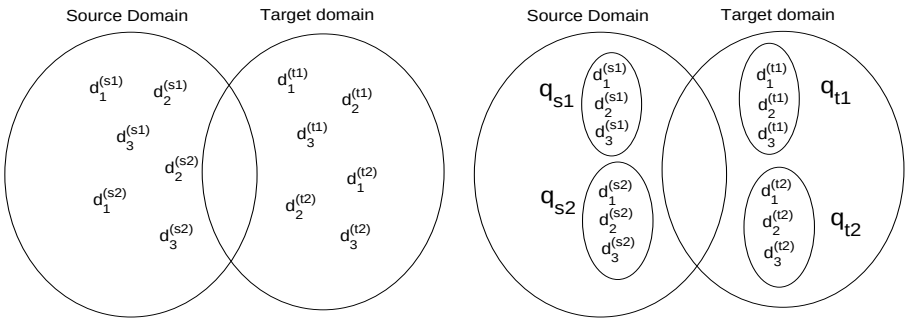
1 Introduction

Learning to rank, which aims at ranking documents in terms of their relevance to user's query, has been widely studied in machine learning and information retrieval communities (Herbrich et al., 2000; Freund et al., 2004; Burges et al., 2005; Yue et al., 2007; Cao et al., 2007; Liu, 2009). In general, large amount of training data need to be annotated by domain experts for achieving better ranking performance. In real applications, however, it is time consuming and expensive to annotate training data for each search domain. To alleviate the lack of training data in the target domain, many researchers have proposed to transfer ranking knowledge from the source domain with plenty of labeled data to the target domain where only a few or no labeled data is available, which is known as ranking model adaptation (Chen et al., 2008a; Chen et al., 2010; Chen et al., 2008b; Geng et al., 2009; Gao et al., 2009).

Intuitively, the more similar an source instance is to the target instances, it is expected to be more useful for cross-domain knowledge transfer. This motivated the popular domain adaptation solution based on instance weighting, which assigns larger weights to those transferable instances so that the model trained on the source domain can adapt more effectively to the target domain (Jiang and Zhai, 2007). Existing instance weighting schemes mainly focus on the adaptation problem for classification (Zadrozny, 2004; Huang et al., 2007; Jiang and Zhai, 2007; Sugiyama et al., 2008).

Although instance weighting scheme may be applied to documents for ranking model adaptation, the difference between classification and learning to rank should be highlighted to take careful consideration. Compared to classification, the learning object for ranking is essentially a query, which contains a list of document instances each with a relevance judgement. Recently, researchers proposed listwise ranking algorithms (Yue et al., 2007; Cao et al., 2007) to take the whole query as a learning object. The benchmark evaluation showed that list-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 112–122,
Portland, Oregon, June 19-24, 2011.



(a) Instance based weighting (b) Query based weighting

Figure 1: The information about which document instances belong to the same query is lost in document instance weighting scheme. To avoid losing this information, query weighting takes the query as a whole and directly measures its importance.

wise approach significantly outperformed pointwise approach, which takes each document instance as independent learning object, as well as pairwise approach, which concentrates learning on the order of a pair of documents (Liu, 2009). Inspired by the principle of listwise approach, we hypothesize that the importance weighting for ranking model adaptation could be done better at query level rather than document level.

Figure 1 demonstrates the difference between instance weighting and query weighting, where there are two queries q_{s1} and q_{s2} in the source domain and q_{t1} and q_{t2} in the target domain, respectively, and each query has three retrieved documents. In Figure 1(a), source and target domains are represented as a bag of document instances. It is worth noting that the information about which document instances belong to the same query is lost. To avoid this information loss, query weighting scheme shown as Figure 1(b) directly measures importance weight at query level.

Instance weighting makes the importance estimation of document instances inaccurate when documents of the same source query are similar to the documents from different target queries. Take Figure 2 as a toy example, where the document instance is represented as a feature vector with four features. No matter what weighting schemes are used, it makes sense to assign high weights to source queries q_{s1} and q_{s2} because they are similar to target queries q_{t1} and q_{t2}, respectively. Meanwhile, the source query q_{s3} should be weighted lower because

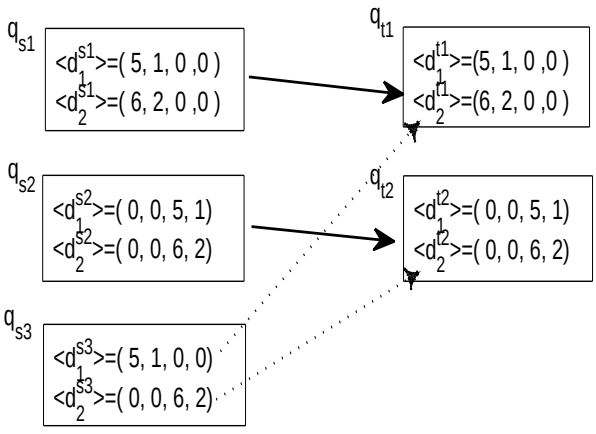


Figure 2: A toy example showing the problem of document instance weighting scheme.

it's not quite similar to any of q_{t1} and q_{t2} at query level, meaning that the ranking knowledge from q_{s3} is different from that of q_{t1} and q_{t2} and thus less useful for the transfer to the target domain. Unfortunately, the three source queries q_{s1}, q_{s2} and q_{s3} would be weighted equally by document instance weighting scheme. The reason is that all of their documents are similar to the two document instances in target domain despite the fact that the documents of q_{s3} correspond to their counterparts from *different* target queries.

Therefore, we should consider the source query as a whole and directly measure the query importance. However, it's not trivial to directly estimate

a query's weight because a query is essentially provided as a matrix where each row represents a vector of document features. In this work, we present two simple but very effective approaches attempting to resolve the problem from distinct perspectives: (1) we compress each query into a *query feature vector* by aggregating all of its document instances, and then conduct query weighting on these query feature vectors; (2) we measure the similarity between the source query and each target query one by one, and then combine these *fine-grained* similarity values to calculate its importance to the target domain.

2 Instance Weighting Scheme Review

The basic idea of instance weighting is to put larger weights on source instances which are more similar to target domain. As a result, the key problem is how to accurately estimate the instance's weight indicating its importance to target domain. (Jiang and Zhai, 2007) used a small number of *labeled* data from target domain to weight source instances. Recently, some researchers proposed to weight source instance only using *unlabeled* target instances (Shimodaira, 2000; Sugiyama et al., 2008; Huang et al., 2007; Zadrozny, 2004; Gao et al., 2010). In this work, we also focus on weighting source queries only using *unlabeled* target queries.

(Gao et al., 2010; Ben-David et al., 2010) proposed to use a classification hyperplane to separate source instances from target instances. With the domain separator, the probability that a source instance is classified to target domain can be used as the importance weight. Other instance weighting methods were proposed for the sample selection bias or covariate shift in the more general setting of classifier learning (Shimodaira, 2000; Sugiyama et al., 2008; Huang et al., 2007; Zadrozny, 2004). (Sugiyama et al., 2008) used a natural model selection procedure, referred to as Kullback-Leibler divergence Importance Estimation Procedure (KLIEP), for automatically tuning parameters, and showed that its importance estimation was more accurate. The main idea is to directly estimate the density function ratio of target distribution $p_t(x)$ to source distribution $p_s(x)$, i.e. $w(x) = \frac{p_t(x)}{p_s(x)}$. Then model $w(x)$ can be used to estimate the importance of source instances. Model parameters were computed with a linear model by minimizing the KL-divergence from $p_t(x)$ to its estimator $\hat{p}_t(x)$. Since $\hat{p}_t(x) = \hat{w}(x)p_s(x)$, the ultimate objective only contains model $\hat{w}(x)$.

For using instance weighting in pairwise ranking algorithms, the weights of document instances should be transformed into those of document pairs (Gao et al., 2010). Given a pair of documents $\langle x_i, x_j \rangle$ and their weights w_i and w_j, the pairwise weight w_{ij} could be estimated probabilistically as $w_i * w_j$. To consider query factor, query weight was further estimated as the average value of the weights over all the pairs, i.e., $w_q = \frac{1}{M} \sum_{i,j} w_{ij}$, where M is the number of pairs in query q. Additionally, to take the advantage of both query and document information, a probabilistic weighting for $\langle x_i, x_j \rangle$ was modeled by $w_q * w_{ij}$. Through the transformation, instance weighting schemes for classification can be applied to ranking model adaptation.

3 Query Weighting

In this section, we extend instance weighting to directly estimate query importance for more effective ranking model adaptation. We present two query weighting methods from different perspectives. Note that although our methods are based on domain separator scheme, other instance weighting schemes such as KLIEP (Sugiyama et al., 2008) can also be extended similarly.

3.1 Query Weighting by Document Feature Aggregation

Our first query weighting method is inspired by the recent work on local learning for ranking (Geng et al., 2008; Banerjee et al., 2009). The query can be compressed into a query feature vector, where each feature value is obtained by the aggregate of its corresponding features of all documents in the query. We concatenate two types of aggregates to construct the query feature vector: the mean $\vec{\mu} = \frac{1}{|q|} \sum_{i=1}^{|q|} \vec{f_i}$ and the variance $\vec{\sigma} = \frac{1}{|q|} \sum_{i=1}^{|q|} (\vec{f_i} - \vec{\mu})^2$, where $\vec{f_i}$ is the feature vector of document i and $|q|$ denotes the number of documents in q . Based on the aggregation of documents within each query, we can use a domain separator to directly weight the source queries with the set of queries from both domains.

Given query data sets $D_s = \{q_s^i\}_{i=1}^m$ and $D_t = \{q_t^j\}_{j=1}^n$ respectively from the source and target do-

Algorithm 1 Query Weighting Based on Document Feature Aggregation in the Query

Input:

Queries in the source domain, $D_s = \{q_s^i\}_{i=1}^m$;

Queries in the target domain, $D_t = \{q_t^j\}_{j=1}^n$;

Output:

Importance weights of queries in the source domain, $IW_s = \{W_i\}_{i=1}^m$;

1: $y_s = -1, y_t = +1$;
2: **for** $i = 1; i \leq m; i++$ **do**
3: Calculate the mean vector $\vec{\mu}_i$ and variance vector $\vec{\sigma}_i$ for q_s^i;
4: Add query feature vector $\vec{q}_s^i = (\vec{\mu}_i, \vec{\sigma}_i, y_s)$ to D'_s ;
5: **end for**
6: **for** $j = 1; j \leq n; j++$ **do**
7: Calculate the mean vector $\vec{\mu}_j$ and variance vector $\vec{\sigma}_j$ for q_t^j;
8: Add query feature vector $\vec{q}_t^j = (\vec{\mu}_j, \vec{\sigma}_j, y_t)$ to D'_t;
9: **end for**
10: Find classification hyperplane H_{st} which separates D'_s from D'_t;
11: **for** $i = 1; i \leq m; i++$ **do**
12: Calculate the distance of $\vec{q}_s^i$ to H_{st}, denoted as $\mathcal{L}(\vec{q}_s^i)$;
13: $W_i = P(q_s^i \in D_t) = \frac{1}{1+exp(\alpha * \mathcal{L}(\vec{q}_s^i)+\beta)}$
14: Add W_i to IW_s;
15: **end for**
16: **return** IW_s;

mains, we use algorithm 1 to estimate the probability that the query q_s^i can be classified to D_t, i.e. $P(q_s^i \in D_t)$, which can be used as the importance of q_s^i relative to the target domain. From step 1 to 9, D'_s and D'_t are constructed using query feature vectors from source and target domains. Then, a classification hyperplane H_{st} is used to separate D'_s from D'_t in step 10. The distance of the query feature vector $\vec{q}_s^i$ from H_{st} are transformed to the probability $P(q_s^i \in D_t)$ using a sigmoid function (Platt and Platt, 1999).

3.2 Query Weighting by Comparing Queries across Domains

Although the query feature vector in algorithm 1 can approximate a query by aggregating its documents' features, it potentially fails to capture important feature information due to the averaging effect during the aggregation. For example, the merit of features in some influential documents may be canceled out in the mean-variance calculation, resulting in many distorted feature values in the query feature vector that hurts the accuracy of query classification hyperplane. This urges us to propose another query weighting method from a different perspective of query similarity.

Intuitively, the importance of a source query to the target domain is determined by its overall similarity to every target query. Based on this intuition, we leverage domain separator to measure the similarity between a source query and each one of the target queries, where an individual domain separator is created for each pair of queries. We estimate the weight of a source query using algorithm 2. Note that we assume document instances in the same query are conditionally independent and all queries are independent of each other. In step 3, $D'_{q_s^i}$ is constructed by all the document instances $\{\vec{x}_k\}$ in query q_s^i with the domain label y_s. For each target query q_t^j, we use the classification hyperplane H_{ij} to estimate $P(\vec{x}_k \in D'_{q_t^j})$, i.e. the probability that each document $\vec{x}_k$ of q_s^i is classified into the document set of q_t^j (step 8). Then the similarity between q_s^i and q_t^j is measured by the probability $P(q_s^i \sim q_t^j)$ at step 9. Finally, the probability of q_s^i belonging to the target domain $P(q_s^i \in D_t)$ is calculated at step 11.

It can be expected that algorithm 2 will generate

Algorithm 2 Query Weighting by Comparing Source and Target Queries

Input:

Queries in source domain, $D_s = \{q_s^i\}_{i=1}^m$;

Queries in target domain, $D_t = \{q_t^j\}_{j=1}^n$;

Output:

Importance weights of queries in source domain, $IW_s = \{W_i\}_{i=1}^m$;

$y_s = -1, y_t = +1$;
for $i = 1; i \leq m; i++$ **do**
  Set $D'_{q_s^i}$=$\{\vec{x}_k, y_s)\}_{k=1}^{|q_s^i|}$;
  for $j = 1; j \leq n; j++$ **do**
    Set $D'_{q_t^j}$=$\{\vec{x}_{k'}, y_t)\}_{k'=1}^{|q_t^j|}$;
    Find a classification hyperplane H_{ij} which separates $D'_{q_s^i}$ from $D'_{q_t^j}$;
    For each k, calculate the distance of $\vec{x}_k$ to H_{ij}, denoted as $\mathcal{L}(\vec{x}_k)$;
    For each k, calculate $P(\vec{x}_k \in D'_{q_t^j}) = \frac{1}{1+exp(\alpha * \mathcal{L}(\vec{x}_k)+\beta)}$;
    Calculate $P(q_s^i \sim q_t^j) = \frac{1}{|q_s^i|} \sum_{k=1}^{|q_s^i|} P(\vec{x}_k \in D'_{q_t^j})$;
  end for
  Add $W_i = P(q_s^i \in D_t) = \frac{1}{n} \sum_{j=1}^n P(q_s^i \sim q_t^j)$ to IW_s;
end for
return IW_s;

more precise measures of query similarity by utilizing the more fine-grained classification hyperplane for separating the queries of two domains.

4 Ranking Model Adaptation via Query Weighting

To adapt the source ranking model to the target domain, we need to incorporate query weights into existing ranking algorithms. Note that query weights can be integrated with either pairwise or listwise algorithms. For pairwise algorithms, a straightforward way is to assign the query weight to all the document pairs associated with this query. However, document instance weighting cannot be appropriately utilized in listwise approach. In order to compare query weighting with document instance weighting, we need to fairly apply them for the same approach of ranking. Therefore, we choose pairwise approach to incorporate query weighting. In this section, we extend Ranking SVM (RSVM) (Herbrich et al., 2000; Joachims, 2002) — one of the typical pairwise algorithms for this.

Let's assume there are m queries in the data set of source domain, and for each query q_i there are $\ell(q_i)$ number of meaningful document pairs that can be constructed based on the ground truth rank labels. Given ranking function f, the objective of RSVM is presented as follows:

$$\min \frac{1}{2}||\vec{w}||^2 + C \sum_{i=1}^{m} \sum_{j=1}^{\ell(q_i)} \xi_{ij} \quad (1)$$
$$\text{subject to } z_{ij} * f(\vec{w}, \vec{x}_{q_i}^{j(1)} - \vec{x}_{q_i}^{j(2)}) \geq 1 - \xi_{ij}$$
$$\xi_{ij} \geq 0, i = 1, \ldots, m; j = 1, \ldots, \ell(q_i)$$

where $\vec{x}_{q_i}^{j(1)}$ and $\vec{x}_{q_i}^{j(2)}$ are two documents with different rank label, and $z_{ij} = +1$ if $\vec{x}_{q_i}^{j(1)}$ is labeled more relevant than $\vec{x}_{q_i}^{j(2)}$; or $z_{ij} = -1$ otherwise.

Let $\lambda = \frac{1}{2C}$ and replace ξ_{ij} with Hinge Loss function $(.)^+$, Equation 1 can be turned to the following form:

$$\min \lambda||\vec{w}||^2 + \sum_{i=1}^{m} \sum_{j=1}^{\ell(q_i)} \left(1 - z_{ij} * f(\vec{w}, \vec{x}_{q_i}^{j(1)} - \vec{x}_{q_i}^{j(2)})\right)^+ \quad (2)$$

Let $IW(q_i)$ represent the importance weight of source query q_i. Equation 2 is extended for integrating the query weight into the loss function in a

straightforward way:

$$\min \lambda||\vec{w}||^2 + \sum_{i=1}^{m} IW(q_i) * \sum_{j=1}^{\ell(q_i)} \left(1 - z_{ij} * f(\vec{w}, \vec{x}_{q_i}^{j(1)} - \vec{x}_{q_i}^{j(2)})\right)^+$$

where $IW(.)$ takes any one of the weighting schemes given by algorithm 1 and algorithm 2.

5 Evaluation

We evaluated the proposed two query weighting methods on TREC-2003 and TREC-2004 web track datasets, which were released through LETOR3.0 as a benchmark collection for learning to rank by (Qin et al., 2010). Originally, different query tasks were defined on different parts of data in the collection, which can be considered as different domains for us. Adaptation takes place when ranking tasks are performed by using the models trained on the domains in which they were originally defined to rank the documents in other domains. Our goal is to demonstrate that query weighting can be more effective than the state-of-the-art document instance weighting.

5.1 Datasets and Setup

Three query tasks were defined in TREC-2003 and TREC-2004 web track, which are home page finding (HP), named page finding (NP) and topic distillation (TD) (Voorhees, 2003; Voorhees, 2004). In this dataset, each document instance is represented by 64 features, including 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 number of queries of each task is given in Table 1.

The *baseline* ranking model is an RSVM directly trained on the source domain without using any weighting methods, denoted as *no-weight*. We implemented two weighting measures based on domain separator and Kullback-Leibler divergence, referred to *DS* and *KL*, respectively. In *DS* measure, three document instance weighting methods based on probability principle (Gao et al., 2010) were implemented for comparison, denoted as *doc-pair*, *doc-avg* and *doc-comb* (see Section 2). In *KL* measure, there is no probabilistic meaning for KL weight and the *doc-comb* based on KL is not interpretable, and we only present the results of *doc-pair* and *doc-avg* for KL measure. Our proposed query weighting methods are denoted by *query-aggr* and *query-comp*, corresponding to document feature aggregation in query and query comparison across domains, respectively. All ranking models above were trained only on source domain training data and the labeled data of target domain was just used for testing.

Query Task	TREC 2003	TREC 2004
Topic Distillation	50	75
Home Page finding	150	75
Named Page finding	150	75

Table 1: The number of queries in TREC-2003 and TREC-2004 web track

For training the models efficiently, we implemented RSVM with Stochastic Gradient Descent (SGD) optimizer (Shalev-Shwartz et al., 2007). The reported performance is obtained by five-fold cross validation.

5.2 Experimental Results

The task of HP and NP are more similar to each other whereas HP/NP is rather different from TD (Voorhees, 2003; Voorhees, 2004). Thus, we carried out HP/NP to TD and TD to HP/NP ranking adaptation tasks. Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 1999) is used as the ranking performance measure.

5.2.1 Adaptation from HP/NP to TD

The first set of experiments performed adaptation from HP to TD and NP to TD. The results of MAP are shown in Table 2.

For the DS-based measure, as shown in the table, *query-aggr* works mostly better than *no-weight*,*doc-pair*, *doc-avg* and *doc-comb*, and *query-comp* performs the best among the five weighting methods. T-test on MAP indicates that the improvement of *query-aggr* over *no-weight* is statistically significant on two adaptation tasks while the improvement of document instance weighting over *no-weight* is statistically significant only on one task. All of the improvement of *query-comp* over *no-weight*, *doc-pair*,*doc-avg* and *doc-comb* are statistically significant. This demonstrates the effectiveness of query

Model	Weighting method	HP03 to TD03	HP04 to TD04	NP03 to TD03	NP04 to TD04
	no-weight	0.2508	0.2086	0.1936	0.1756
DS	*doc-pair*	0.2505	0.2042	$0.1982^{\dagger}$	0.1708
	doc-avg	0.2514	0.2019	$0.2122^{\dagger\ddagger}$	0.1716
	doc-comb	0.2562	0.2051	$0.2224^{\dagger\ddagger\sharp}$	0.1793
	query-aggr	0.2573	$\mathbf{0.2106}^{\dagger\ddagger\sharp}$	0.2088	$\mathbf{0.1808}^{\dagger\ddagger\sharp}$
	query-comp	$\mathbf{0.2816}^{\dagger\ddagger\sharp}$	$\mathbf{0.2147}^{\dagger\ddagger\sharp}$	$\mathbf{0.2392}^{\dagger\ddagger\sharp}$	$\mathbf{0.1861}^{\dagger\ddagger\sharp}$
KL	*doc-pair*	0.2521	0.2048	0.1901	0.1761
	doc-avg	0.2534	$0.2127^{\dagger}$	0.1904	0.1777
	doc-comb	-	-	-	-
	query-aggr	0.1890	0.1901	0.1870	0.1643
	query-comp	$0.2548^{\dagger}$	$0.2142^{\dagger}$	$0.2313^{\dagger\ddagger\sharp}$	$0.1807^{\dagger}$

Table 2: Results of MAP for HP/NP to TD adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *doc-pair*, *doc-avg* and *doc-comb*, respectively. Confidence level is set at 95%

weighting compared to document instance weighting.

Furthermore, *query-comp* can perform better than *query-aggr*. The reason is that although document feature aggregation might be a reasonable representation for a set of document instances, it is possible that some information could be lost or distorted in the process of compression. By contrast, more accurate query weights can be achieved by the more fine-grained similarity measure between the source query and all target queries in algorithm 2.

For the KL-based measure, similar observation can be obtained. However, it's obvious that DS-based models can work better than the KL-based. The reason is that KL conducts weighting by density function ratio which is sensitive to the data scale. Specifically, after document feature aggregation, the number of query feature vectors in all adaptation tasks is no more than 150 in source and target domains. It renders the density estimation in *query-aggr* is very inaccurate since the set of samples is too small. As each query contains 1000 documents, they seemed to provide *query-comp* enough samples for achieving reasonable estimation of the density functions in both domains.

5.2.2 Adaptation from TD to HP/NP

To further validate the effectiveness of query weighting, we also conducted adaptation from TD to HP and TD to NP . MAP results with significant test are shown in Table 3.

We can see that document instance weighting schemes including *doc-pair*, *doc-avg* and *doc-comb* can not outperform *no-weight* based on MAP measure. The reason is that each query in TD has 1000 retrieved documents in which 10-15 documents are relevant whereas each query in HP or NP only consists 1-2 relevant documents. Thus, when TD serves as the source domain, it leads to the problem that too many document pairs were generated for training the RSVM model. In this case, a small number of documents that were weighted inaccurately can make significant impact on many number of document pairs. Since query weighting method directly estimates the query importance instead of document instance importance, both *query-aggr* and *query-comp* can avoid such kind of negative influence that is inevitable in the three document instance weighting methods.

5.2.3 The Analysis on Source Query Weights

An interesting problem is which queries in the source domain are assigned high weights and why it's the case. Query weighting assigns each source query with a weight value. Note that it's not meaningful to directly compare absolute weight values between *query-aggr* and *query-comp* because source query weights from distinct weighting methods have different range and scale. However, it is feasible to compare the weights with the same weighting method. Intuitively, if the ranking model learned from a source query can work well in target domain, it should get high weight. According to this intuition, if ranking models $f_{q_s^1}$ and $f_{q_s^2}$ are learned

model	weighting scheme	TD03 to HP03	TD04 to HP04	TD03 to NP03	TD04 to NP04
	no-weight	0.6986	0.6158	0.5053	0.5427
DS	*doc-pair*	0.6588	$0.6235^{\dagger}$	0.4878	0.5212
	doc-avg	0.6654	0.6200	0.4736	0.5035
	doc-comb	0.6932	$0.6214^{\dagger}$	0.4974	0.5077
	query-aggr	$\mathbf{0.7179}^{\dagger\ddagger\sharp}$	$0.6292^{\dagger\ddagger\sharp}$	$\mathbf{0.5198}^{\dagger\ddagger\sharp}$	$\mathbf{0.5551}^{\dagger\ddagger\sharp}$
	query-comp	$\mathbf{0.7297}^{\dagger\ddagger\sharp}$	$\mathbf{0.6499}^{\dagger\ddagger\sharp}$	$\mathbf{0.5203}^{\dagger\ddagger\sharp}$	$\mathbf{0.6541}^{\dagger\ddagger\sharp}$
KL	*doc-pair*	0.6480	0.6107	0.4633	0.5413
	doc-avg	0.6472	0.6132	0.4626	0.5406
	doc-comb	–	–	–	–
	query-aggr	0.6263	0.5929	0.4597	0.4673
	query-comp	$0.6530^{\ddagger\sharp}$	$0.6358^{\dagger\ddagger\sharp}$	0.4726	$0.5559^{\dagger\ddagger\sharp}$

Table 3: Results of MAP for TD to HP/NP adaptation. †, ‡, ♯ and boldface indicate significantly better than *no-weight*, *doc-pair*, *doc-avg* and *doc-comb*, respectively. Confidence level is set as 95%.

from queries q_s^1 and q_s^2 respectively, and $f_{q_s^1}$ performs better than $f_{q_s^2}$, then the source query weight of q_s^1 should be higher than that of q_s^2.

For further analysis, we compare the weight values between each source query pair, for which we trained RSVM on each source query and evaluated the learned model on test data from target domain. Then, the source queries are ranked according to the MAP values obtained by their corresponding ranking models. The order is denoted as R_{map}. Meanwhile, the source queries are also ranked with respect to their weights estimated by DS-based measure, and the order is denoted as R_{weight}. We hope R_{weight} is correlated as positively as possible with R_{map}. For comparison, we also ranked these queries according to randomly generated query weights, which is denoted as *query-rand* in addition to *query-aggr* and *query-comp*. The Kendall's $\tau = \frac{P-Q}{P+Q}$ is used to measure the correlation (Kendall, 1970), where P is the number of concordant query pairs and Q is the number of discordant pairs. It's noted that τ's range is from -1 to 1, and the larger value means the two ranking is better correlated. The Kendall's τ by different weighting methods are given in Table 4 and 5.

We find that R_{weight} produced by *query-aggr* and *query-comp* are all positively correlated with R_{map} and clearly the orders generated by *query-comp* are more positive than those by *query-aggr*. This is another explanation why *query-comp* outperforms *query-aggr*. Furthermore, both are far better than *query-rand* because the R_{weight} by *query-rand* is actually independent of R_{map}.

weighting	TD03 to HP03	TD04 to HP04
doc-pair	28,835 secs	21,640 secs
query-aggr	182 secs	123 secs
query-comp	15,056 secs	10,081 secs

Table 6: The efficiency of weighting in seconds.

5.2.4 Efficiency

In the situation where there are large scale data in source and target domains, how to efficiently weight a source query is another interesting problem. Without the loss of generality, we reported the weighting time of *doc-pair*, *query-aggr* and *query-comp* from adaptation from TD to HP using DS measure. As *doc-avg* and *doc-comb* are derived from *doc-pair*, their efficiency is equivalent to *doc-pair*.

As shown in table 6, *query-aggr* can efficiently weight query using query feature vector. The reason is two-fold: one is the operation of query document aggregation can be done very fast, and the other is there are 1000 documents in each query of TD or HP, which means that the compression ratio is 1000:1. Thus, the domain separator can be found quickly. In addition, *query-comp* is more efficient than *doc-pair* because *doc-pair* needs too much time to find the separator using all instances from source and target domain. And *query-comp* uses a divide-and-conquer method to measure the similarity of source query to each target query, and then efficiently combine these

Weighting method	HP03 to TD03	HP04 to TD04	NP03 to TD03	NP04 to TD04
query-aggr	0.0906	0.0280	0.0247	0.0525
query-comp	0.1001	0.0804	0.0711	0.1737
query-rand	0.0041	0.0008	-0.0127	0.0163

Table 4: The Kendall's τ of R_{weight} and R_{map} in HP/NP to TD adaptation.

Weighting method	TD03 to HP03	TD04 to HP04	TD03 to NP03	TD04 to NP04
query-aggr	0.1172	0.0121	0.0574	0.0464
query-comp	0.1304	0.1393	0.1586	0.0545
query-rand	-0.0291	0.0022	0.0161	-0.0262

Table 5: The Kendall's τ of R_{weight} and R_{map} in TD to HP/NP adaptation.

fine-grained similarity values.

6 Related Work

Cross-domain knowledge transfer has became an important topic in machine learning and natural language processing (Ben-David et al., 2010; Jiang and Zhai, 2007; Blitzer et al., 2006; Daumé III and Marcu, 2006). (Blitzer et al., 2006) proposed model adaptation using pivot features to build structural feature correspondence in two domains. (Pan et al., 2009) proposed to seek a common features space to reduce the distribution difference between the source and target domain. (Daumé III and Marcu, 2006) assumed training instances were generated from source domain, target domain and cross-domain distributions, and estimated the parameter for the mixture distribution.

Recently, domain adaptation in learning to rank received more and more attentions due to the lack of training data in new search domains. Existing ranking adaptation approaches can be grouped into feature-based (Geng et al., 2009; Chen et al., 2008b; Wang et al., 2009; Gao et al., 2009) and instance-based (Chen et al., 2010; Chen et al., 2008a; Gao et al., 2010) approaches. In (Geng et al., 2009; Chen et al., 2008b), the parameters of ranking model trained on the source domain was adjusted with the small set of labeled data in the target domain. (Wang et al., 2009) aimed at ranking adaptation in heterogeneous domains. (Gao et al., 2009) learned ranking models on the source and target domains independently, and then constructed a stronger model by interpolating the two models. (Chen et al., 2010; Chen et al., 2008a) weighted source instances by using small amount of labeled data in the target domain. (Gao et al., 2010) studied instance weighting based on domain separator for learning to rank by only using training data from source domain. In this work, we propose to directly measure the query importance instead of document instance importance by considering information at both levels.

7 Conclusion

We introduced two simple yet effective query weighting methods for ranking model adaptation. The first represents a set of document instances within the same query as a query feature vector, and then directly measure the source query importance to the target domain. The second measures the similarity between a source query and each target query, and then combine the fine-grained similarity values to estimate its importance to target domain. We evaluated our approaches on LETOR3.0 dataset for ranking adaptation and found that: (1) the first method efficiently estimate query weights, and can outperform the document instance weighting but some information is lost during the aggregation; (2) the second method consistently and significantly outperforms document instance weighting.

8 Acknowledgement

P. Cai and A. Zhou are supported by NSFC (No. 60925008) and 973 program (No. 2010CB731402). W. Gao and K.-F. Wong are supported by national 863 program (No. 2009AA01Z150). We also thank anonymous reviewers for their helpful comments.

References

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*.

Somnath Banerjee, Avinava Dubey, Jinesh Machchhar, and Soumen Chakrabarti. 2009. Efficient and accurate local learning for ranking. In *SIGIR workshop : Learning to rank for information retrieval*, pages 1–8.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

John Blitzer, Ryan Mcdonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of ICML*, pages 89–96.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of ICML*, pages 129 – 136.

Depin Chen, Jun Yan, Gang Wang, Yan Xiong, Weiguo Fan, and Zheng Chen. 2008a. Transrank: A novel algorithm for transfer of rank learning. In *Proceedings of ICDM Workshops*, pages 106–115.

Keke Chen, Rongqing Lu, C.K. Wong, Gordon Sun, Larry Heck, and Belle Tseng. 2008b. Trada: Tree based ranking function adaptation. In *Proceedings of CIKM*.

Depin Chen, Yan Xiong, Jun Yan, Gui-Rong Xue, Gang Wang, and Zheng Chen. 2010. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126.

Y. Freund, R. Iyer, R. Schapire, and Y. Singer. 2004. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Jianfeng Gao, Qiang Wu, Chris Burges, Krysta Svore, Yi Su, Nazan Khan, Shalin Shah, and Hongyan Zhou. 2009. Model adaptation via model interpolation and boosting for web search ranking. In *Proceedings of EMNLP*.

Wei Gao, Peng Cai, Kam Fai Wong, and Aoying Zhou. 2010. Learning to rank only using training data from related domain. In *Proceedings of SIGIR*, pages 162–169.

Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. 2008. Query dependent ranking using k-nearest neighbor. In *Proceedings of SIGIR*, pages 115–122.

Bo Geng, Linjun Yang, Chao Xu, and Xian-Sheng Hua. 2009. Ranking model adaptation for domain-specific search. In *Proceedings of CIKM*.

R. Herbrich, T. Graepel, and K. Obermayer. 2000. *Large Margin Rank Boundaries for Ordinal Regression*. MIT Press, Cambridge.

Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. 2007. Correcting sample selection bias by unlabeled data. In *Proceedings of NIPS*, pages 601–608.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of ACL*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD*, pages 133–142.

Maurice Kendall. 1970. *Rank Correlation Methods*. Griffin.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2009. Domain adaptation via transfer component analysis. In *Proceedings of IJCAI*, pages 1187–1192.

John C. Platt and John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. 2008. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of NIPS*, pages 1433–1440.

Ellen M. Voorhees. 2003. Overview of trec 2003. In *Proceedings of TREC-2003*, pages 1–13.

Ellen M. Voorhees. 2004. Overview of trec 2004. In *Proceedings of TREC-2004*, pages 1–12.

Bo Wang, Jie Tang, Wei Fan, Songcan Chen, Zi Yang, and Yanzhu Liu. 2009. Heterogeneous cross domain ranking in latent space. In *Proceedings of CIKM*.

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. 2007. A support vector method for optimizing average precision. In *Proceedings of SIGIR*, pages 271–278.

Bianca Zadrozny Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of ICML*, pages 325–332.