Applying Co-training to Clickthrough Data for Search Engine Adaptation*

Qingzhao Tan Xiaoyong Chai Wilfred Ng Dik-Lun Lee

Department of Computer Science
The Hong Kong University of Science and Technology
{ttqqzz, carnamel, wilfred, dlee}@cs.ust.hk

Abstract. The information on the World Wide Web is growing without bound. Users may have very diversified preferences in the pages they target through a search engine. It is therefore a challenging task to adapt a search engine to suit the needs of a particular community of users who share similar interests. In this paper, we propose a new algorithm, Ranking SVM in a Co-training Framework (RSCF). Essentially, the RSCF algorithm takes the clickthrough data containing the items in the search result that have been clicked on by a user as an input, and generates adaptive rankers as an output. By analyzing the clickthrough data, RSCF first categorizes the data as the labelled data set, which contains the items that have been scanned already, and the unlabelled data set, which contains the items that have not yet been scanned. The labelled data is then augmented with unlabelled data to obtain a larger data set for training the rankers. We demonstrate that the RSCF algorithm produces better ranking results than the standard Ranking SVM algorithm. Based on RSCF we develop a metasearch engine that comprises MSNSearch, Wisenut, and Overture, and carry out an online experiment to show that our metasearch engine outperforms Google.

Keywords: Clickthrough data, co-training, adaptation, search engine

1 Introduction

Users may have very diversified preferences in the pages they target through a search engine. It is therefore a challenging task to adapt a search engine to suit the needs of a particular community of users who share similar interests.

Some previous approaches for optimizing search engines require training data generated from users' explicit relevance judgments on search results [1–3]. Users are usually unwilling to give such feedback because of privacy concerns and the extra effort required by them. To overcome this problem, we propose using clickthrough data, which is a log kept by the search engine on the queries submitted by users, the returned result items, and the items, if any, clicked on by the users, as an implicit relevance feedback on the search results.

Formally, clickthrough data can be denoted as a triplet (q, r, c), where q is the input query consisting of a set of keywords, r is a list of ranked links, $(l_1, \ldots, l_n)$, and c is the set of links that the user has clicked on. Figure 1 illustrates the submitted query q "Biometrics Research" and the returned list of the ranked result. In this

* This work is supported in part by grants from the Research Grant Council of Hong Kong, Grant No HKUST6079/01E, DAG01/02.EG05, and HKUST6185/02E.

example we assume that a user scans the ranking presented from top to bottom. The three links l_1, l_7 and l_{10} are bold, which means that they have been clicked on by the user.

Links	Information of web pages in the search results
l_1 (clicked)	Biometrics Research Page Provides an overview of the technology... **biometrics.cse.msu.edu**
l_2	National Cancer Institute - Biometric Research Branch Part of the NCI's Division of Cancer Treatment and Diagnosis... linus.nci.nih.gov/ brb
l_3	International Biometric Group Private firm provides consulting services, biometric research, and ... www.biometricgroup.com
l_4	Microsoft Research - Vision Technology Find out about the computer vision research groups at Microsoft... research.microsoft.com/research/vision
l_5	Forest Biometrics Research Institute Forest Biometrics Research Institute University of Montana... www.forestbiometrics.com/Institute.htm
l_6	Signal Processing Research Centre Queensland University of Technology in Australia presents ... www.sprc.qut.edu.au/research/fingerprint.html
l_7 (clicked)	Research: Biometrics Network World articles and Internet resources about ... **www.nwfusion.com/research/biometrics.html**
l_8	Biometrics: Overview An Overview of Biometrics refers to the automatic ... biometrics.cse.msu.edu/info.html
l_9	TeKey Research Group Download trial version of its software used for image processing and ... www.tekey.com
l_{10} (clicked)	Biometrics — Research Areas — IRL A sentence describing the contents and subject of the ... **www.research.ibm.com/irl/projects/biometrics**

Fig. 1. The clickthrough data for the query "Biometrics Research"

Analyzing clickthrough data is a useful means to understand users' target preference in the returned search results, since it conveys *partial* relative relevance judgments on the links that a user has browsed through [4]. However, if we consider a collection of clickthrough data generated from a large number of users, the clickthrough data may be too diversified for inferring the best results for a query across the entire user group. On the other hand, we observe that for a particular community of users who share interests in the same domain (e.g., Computer Science), their behavior is relatively similar and the inference of preference can be much more reliable.

There are other proposed learning retrieval functions using clickthrough data. In [5], clickthrough data was used to optimize the ranking in search engines. However, the semantics of the learning process and its results were not clear. Based on these pioneering works, Joachims proposed the RSVM algorithm which uses clickthrough data to optimize the performance of a retrieval function [6]. The limitation

of Joachims' algorithm is that it requires a large set of training data to make the algorithm effective.

In this paper, we develop a new algorithm, which we call Ranking SVM in a Co-training Framework (RSCF). Essentially, the RSCF algorithm takes the clickthrough data as an input and generates adaptive rankers as an output in a learning process. It is an enhancement of the RSVM [7]. RSCF incorporates into RSVM a co-training framework [8] to make the learning process efficient even when the amount of training data set is relatively small and sparse.

RSCF analyzes the clickthrough data extracted from the log files and then categorizes the data as the labelled data set, which contains the search items that have been scanned by users, and the unlabelled data set, which contains the data items that have not yet been scanned. It then augments the labelled data with the unlabelled data to obtain a larger data set for training the rankers. As evidenced from offline experiments, the RSCF algorithm produces better ranking results than the standard RSVM algorithm in terms of prediction error. Based on the RSCF setting, we develop a metasearch engine that comprises MSNsearch, Wisenut, and Overture[1], and carry out an online experiment. Notably, the three search engine components used in the metasearch engine prototype are conceived to be weaker than Google[2], but our online experiment shows that our metasearch engine outperforms Google in retrieval quality.

The rest of this paper is organized as follows. In Section 2, we present the RSCF algorithm. In section 3, we demonstrate in offline and online experiments that the RSCF algorithm improves retrieval quality. We give our concluding remarks in Section 4.

2 The Ranking SVM in Co-Training Framework

In this section we analyze clickthrough data via the RSCF algorithm. RSCF addresses the problem that the training set of preference feedback extracted from a single query is relatively small.

We observe in Figure 1 that the user probably scanned l_2, l_3, l_4, l_5, and l_6 before clicking on l_7, i.e., he has made the decision not to click on l_2 to l_6. Therefore, l_7 is more relevant than other links according to the user's preference. In other words, l_7 should rank ahead of these five links in the target ranking. Similarly, l_{10} should rank ahead of l_2, l_3, l_4, l_5, l_6, l_8, and l_9. We now denote the ranking extracted from clickthrough data by r'. It is straightforward to check that the three sets of preference pairs according to the three clicks l_1, l_7 and l_{10} can be obtained as shown in Figure 2. These three sets represent the relevance judgments collectively, where some links are incomparable (e.g., l_1, l_7 and l_{10} are incomparable when paired with respect to $<_{r'}$). Now, let r^* be the target ranking in the search result of a submitted query. Although r^* is optimal to the user, it is not fully observable in practice. However, we are able to obtain r' from clickthrough data, which is in fact a subset of r^*. Given the training set $(q_1, r'_1), (q_2, r'_2), \ldots, (q_n, r'_n)$, we aim at finding a rank that holds as many preference feedback pairs in r' as possible. First, by extracting a feature

[1] At the time of this writing, MSNsearch (website at http://www.msnsearch.com/) was powered by Inktomi, Wisenut (website at http://www.wisenut.com/) was a new but growing search engine, and Overture (website at http://www.overture.com) ranked results based on the prices paid by the sponsors on the results.

[2] Website at http://www.google.com.

Set of preference pairs arising from l_1	Set of preference pairs arising from l_7	Set of preference pairs arising from l_{10}
Empty Set	$l_7 <_{r'} l_2$	$l_{10} <_{r'} l_2$
	$l_7 <_{r'} l_3$	$l_{10} <_{r'} l_3$
	$l_7 <_{r'} l_4$	$l_{10} <_{r'} l_4$
	$l_7 <_{r'} l_5$	$l_{10} <_{r'} l_5$
	$l_7 <_{r'} l_6$	$l_{10} <_{r'} l_6$
		$l_{10} <_{r'} l_8$
		$l_{10} <_{r'} l_9$

Fig. 2. Sets of preference pairs derived from clickthrough data

vector, we can rank the documents in the search result by giving different weight to the features. Then we find a weight vector $\overrightarrow{\omega}$ that makes the set of inequalities given in (1) hold for $1 \leq k \leq n$.

$$\forall (d_i, d_j) \in r'_k : \overrightarrow{\omega}\phi(q_k, d_i) > \overrightarrow{\omega}\phi(q_k, d_j) \tag{1}$$

Here $(d_i, d_j) \in r'_k$ is a document pair which corresponds to the preference feedback pair $(l_i <_{r'_k} l_j)$ with respect to the submitted query q_k, $\phi(q_k, d_i)$ is a mapping that maps q_k onto a sequence of features (or a *feature vector*) that describes the match between q_k and d_i. Figure 3 illustrates how the weight vector $\overrightarrow{\omega}$ determines the ordering of the three documents, d_1, d_2, and d_3, in two dimensions. The documents are ordered as (d_1, d_2, d_3) according to $\overrightarrow{\omega_1}$, and as (d_2, d_1, d_3) according to $\overrightarrow{\omega_2}$. The former is better than the latter if the target ranking is $d_1 <_{r^*} d_2 <_{r^*} d_3$.

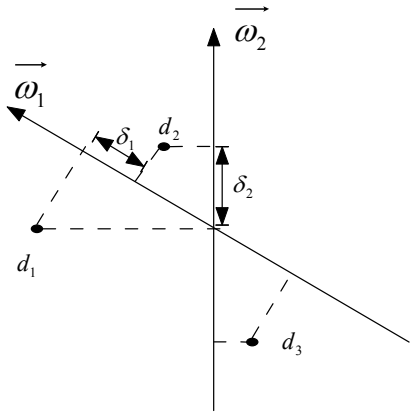

Fig. 3. Ranking d_1, d_2, and d_3 according to the weight vectors $\overrightarrow{\omega_1}$ and $\overrightarrow{\omega_2}$

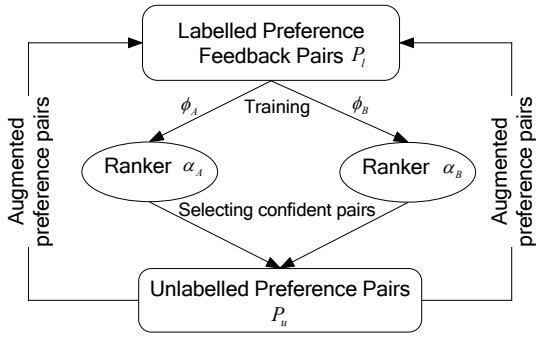

Fig. 4. The underlying idea of RSCF algorithm

The problem of solving $\overrightarrow{\omega}$ using the set of inequalities given in (1) is *NP-hard*. However, an approximated solution can be obtained by introducing a non-negative *slack variable* ξ_{ijk} [7] to tolerate some ranking errors. Recall that r'_k is the subset of the target ranking r^*_k for the search result of the query q_k. Algorithm 1 outlines the RSVM algorithm based on the approximation. The basic idea is that if we consider that δ is the distance between the two closest projected documents, then the larger the value of δ, the more definite the ranking, and hence the better the quality of the training result. Thus, if there are several weight vectors that are able to make all the rankings subject to the condition mentioned in the RSVM algorithm, we will

choose the one that can maximize margin δ. Minimizing $\frac{1}{2}\overrightarrow{\omega} \cdot \overrightarrow{\omega}$ in Algorithm 1 can be viewed as maximizing margin δ. In addition, minimizing $\Sigma\xi_{ijk}$ can be viewed as minimizing the ranking errors. Parameter C is introduced here to allow a trade-off between the margin size and the training errors in Algorithm 1.

Algorithm 1 RSVM Algorithm

Input: r'_k $(1 \leq k \leq n)$ from the set of labelled preference pairs
Procedure:
Minimize: $V(\overrightarrow{\omega}, \xi) = \frac{1}{2}\overrightarrow{\omega} \cdot \overrightarrow{\omega} + C\Sigma\xi_{ijk}$;
Subject to: for all i, j, and k,
$\forall(d_i, d_j) \in r'_k : \overrightarrow{\omega}\phi(q_k, d_i) > \overrightarrow{\omega}\phi(q_k, d_j) + 1 - \xi_{ijk}$; $\xi_{ijk} \geq 0$;
Output: $\overrightarrow{\omega}$

The key idea of co-training is to augment the labelled data with the unlabelled data when the labelled training data are limited and sparse. As the training data set is enlarged, the classification errors are significantly decreased. In the RSCF algorithm, we enhance the RSVM algorithm and incorporate the co-training framework. The feature vector $\phi(q, d)$ describing the match between the query q and the document d is first divided into two feature subvectors, $\phi_1(q, d)$ and $\phi_2(q, d)$, which will be discussed in detail in Section 3.3. Then, the two rankers, α_A and α_B, are incrementally built over these two feature subvectors. Both rankers use the RSVM algorithm to learn weight vectors. Each ranker is initialized with a few labelled preference pairs extracted from the clickthrough data (e.g., (l_7, l_3) in Figure 1). In each iteration of co-training, each ranker chooses several preference pairs (e.g., (l_9, l_8) in Figure 1) from the unlabelled data set and add them to the labelled data set. The chosen pairs are those with the highest ranking confidence as given by the underlying rankers. Then, each ranker is rebuilt from the augmented labelled set. In the next iteration, the new rankers are used to rank the unlabelled preference pairs again. The ranking preference pairs process and the building rankers process repeat until all unlabelled preference pairs are labelled or a terminating criterion is satisfied. Figure 4 illustrates the process and the underlying idea of the algorithm.

The guideline for partitioning the feature vector $\phi(q, d)$ is that after the partition each subvector must be independent and sufficient for the later ranking. In general, the number of rankers used in the co-training framework can be more than two. However, since clickthrough data only contains simple actions of the search engine users, it is difficult to identify enough features from the clickthrough data to generate more than two feature subvectors that are rich enough to train the corresponding rankers. Even if the feature vector is of large dimension, we still should use some feature selection algorithm to eliminate some unimportant features and to reduce the dimension of the feature vector so that the training process can be more efficient. In our study we choose two rankers for running the co-training algorithm.

We now introduce the parameter *prediction error*, which is a common criterion used to evaluate the performance of a classifier in *Machine Learning*. We use it to evaluate the performance of a ranker in our experiments. Let us call a link pair l_i and l_j $(l_i \neq l_j)$ *disconcordant*, if the pair has different rankings in two given lists [6]. Given a ranker α being trained in RSVM, the *prediction error* of α is defined as the percentage of link pairs in the original test data set that are *disconcordant* according to α after an iteration. For example, if there are 50 pairs of links in the original training data set and among them 15 pairs are *disconcordant* links according to α after some iterations, the prediction error of this iteration would be 0.3. We also need

to define another new concept, *prediction difference* Δ, as the terminating criterion used in the RSCF algorithm. Δ is defined as the percentage of *disconcordant* links between α_A and α_B. For example, α_A and α_B, respectively, selects 10 preference pairs from the unlabelled preference pairs and add them into the labelled data set. If five selections from α_A are ranked differently according to α_B, then Δ is equal to 0.5.

The RSCF algorithm is presented in Algorithm 2 . At each iteration those unlabelled preference pairs that have the largest difference between their projection on $\overrightarrow{\omega}$ (i.e., the most confident ones) are selected and then added into the preference pair set. We finally obtain one enlarged set of labelled preference feedback pairs P_l and two rankers, α_A and α_B, as an output of the algorithm. We also set the threshold τ for Δ as an input to the algorithm and compute Δ at the end of each iteration. When Δ reaches τ, the whole process will be terminated. Using the P_l output from Algorithm 2, we are able to obtain a final ranker α_C by the RSVM algorithm. The ranker α_C predicts better relevance judgment for new link pairs than α_R, which is the ranker obtained from standard RSVM, as will be shown in our experiments.

Algorithm 2 RSCF Algorithm

Input: P_l: An initial set of labelled preference pairs;
P_u: An initial set of unlabelled preference pairs;
τ: The threshold for Δ;

Procedure:

1: **while** there exist preference pairs without labels and $\Delta < \tau$ **do**
2: Use RSVM to build α_A using features in ϕ_1 of P_l;
3: Use RSVM to build α_B using features in ϕ_2 of P_l;
4: Select the most confident unlabelled preference pairs from P_u according to α_A and add them to P_l;
5: Select the most confident unlabelled preference pairs from P_u according to α_B and add them to P_l;
6: Compute Δ;
7: **end while**

Output: P_l, α_A and α_B.

3 Experimental Investigation

In order to provide a testbed for comparison, we implement a RSCF-based metasearch engine that combines the results of MSNsearch, Wisenut and Overture. Then we exploit the *unbiased experiment setup* designed in [4] and run the experiment on the RSCF metasearch engine as follows. When the user types a query into a unified interface, the query is sent to the three search engines. The top 10 links obtained from each search engine are combined into one round-robin list. Note that if the same result is returned by more than one search engine, we keep only one of them. Then the links together with titles and abstracts of the documents are displayed to the user in a uniform presentation style. As such, the user cannot tell which search engine contributes to a particular link.

3.1 Feature Extraction

In the experiments, we extract 16 features and apply them to define the feature vector mapping $\phi(q, d)$. Our approach of feature extraction is a fundamental technique used

in IR. We further classify these features into two categories, namely, *Ranking Features* and *Similarity Features*. We regard a query q in our context as a set of keywords submitted to a search engine by the user, that is, $q = \{w_1, w_2, \ldots, w_{n_q}\}$, where n_q denotes the number of keywords in q.

1. Ranking Features (There are 12 binary features in total).
   Let $E \in \{M, W, O\}$ (M stands for MSNsearch, W stands for Wisenut, and O stands for Overture) and $T \in \{1, 3, 5, 10\}$. We define the *Ranking Features* as follows:
   $$Rank(E,T) = \begin{cases} 1 & \text{if } d \text{ is ranked top } T \text{ in } E; \\ 0 & \text{otherwise.} \end{cases}$$
2. Similarity Features (There are 4 features in total).
   - *Sim_U(q,l)*:
     We define the similarity between the query q and the URL of the retrieved document d as follows:
     $$Sim\_U(q,l) = \begin{cases} 1 & \text{if any word in } q \text{ appears in the URL;} \\ 0 & \text{otherwise.} \end{cases}$$
   - *Sim_T(q,t)*:
     Let t be the set of terms appearing in the title and N be the total number of *non-stop words* occurring in t. Here, *stop words* are those words having no meaning from the point of view of searching, such as the articles in English, otherwise words are *non-stop words*. We denote P_+ the frequency of the terms in t belonging to q (positive samples), and P_- the frequency of the terms in t not belonging to q (negative samples). We define the similarity between the query q and the title t of the retrieved document d as follows:
     $$Sim\_T(q,t) = \begin{cases} \log N & \text{if } \forall w_i \in t,\ w_i \in q; \\ -\log N & \text{if } \forall w_i \in t,\ w_i \notin q; \\ \frac{1}{2} \log \frac{(1-P_-)P_+}{(1-P_+)P_-} & \text{otherwise.} \end{cases}$$
     There are three exclusive cases in the above formula. In the first case, all the terms in the title are query words. In the second case, none of the terms in the title is among the query words. In the third case, we use the log ratio measurement which takes the frequency of both positive and negative samples into consideration.
   - *Sim_C(q,a)*:
     Let $|q \cap a|$ be the number of words in a query q that also appear in the abstract a. We define the *covering similarity* between the query q and the abstract a of the retrieved document d as follows:
     $$Sim\_C(q,a) = \frac{|q \cap a|}{|q|}$$
   - *Sim_G(q,a)*:
     Let $count(w_i \ in \ a)$ be the number of times a particular keyword w_i occurring in the abstract and $count(q \ in \ a)$ be the number of times the query (i.e., *all the keywords as a whole*) occurs in the abstract. We define the *group-occurring similarity* between the query q and the abstract a of the retrieved document d as follows:
     $$Sim\_G(q,a) = \frac{n_q \cdot count(q \ in \ a)}{\Sigma_{w_i \in q} count(w_i \ in \ a)}$$

We now define the feature vector $\phi(q, d)$ by using all the above extracted features as the vector components as given below:

$$(Rank(M, 1), Rank(M, 3), \ldots, Sim\_U(q, l), \ldots, Sim\_G(q, a))$$

The following example helps to illustrate the use of the above-mentioned features in the feature vector.

Example 1 Suppose the query "Biometrics Research" is submitted to our metasearch engine. We assume that the document information shown in Figure 5 is an item in the search result, which is ranked 5, 11 and 3, respectively, by the underlying search engines of MSNsearch, Overture, and Wisenut.

t	Forest **Biometrics Research** Institute
α	Forest **Biometrics Research** Institute University of Montana, Missoula Founded in 2002 to advance **research**, education in forest **biometrics**. Background - Institute - **Research** ...
URL	*www.forest<u>biometrics</u>.com/Institute.html*

Fig. 5. The information of a retrieved document

We now have $q = \{Biometrics, Research\}$ and four of the ranking features of d that related to M are given as follows: $Rank(M, 1) = 0$, $Rank(M, 3) = 0$, $Rank(M, 5) = 1$ and $Rank(M, 10) = 1$. Similarly, the other eight ranking features related to O and W are given as follows: $Rank(O, 1) = 0$, $Rank(O, 3) = 0$, $Rank(O, 5) = 0$, $Rank(O, 10) = 0$, $Rank(W, 1) = 0$, $Rank(W, 3) = 1$, $Rank(W, 5) = 1$, and $Rank(W, 10) = 1$.

As indicated by the underlined word "biometrics" in URL, which is one of the two query words that appears in the URL, it makes $Sim\_U(q, l) = 1$. As for the feature $Sim\_T(q, t)$, since there are two words, "Biometrics" and "Research", in the title belonging to q, and also two words, "Forest" and "Institute", not belonging to q, $P_+ = \frac{2}{4}$, $P_- = \frac{2}{4}$, and $Sim\_T(q, t) = \log \frac{(1-P_-)P_+}{(1-P_+)P_-} = 0$. It is also easy to arrive at $Sim\_C(q, a) = 1$, since in this example $|q| = 2$ and $|q \cap a| = 2$ (i.e., both the words "Biometrics" and "Research" appear in the abstract). In the abstract, the word "Biometrics" occurs twice, "Research" occurs three times and the phrase "Biometrics Research" as a whole occurs only once. So we have $Sim\_G(q, a) = \frac{2 \cdot 1}{2+3} = 0.4$.

Therefore, the feature vector is given as follows:

$$\phi(q, d) = (0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0.4).$$

3.2 Experiment Data

We collect the sample data, P_l and P_u, from the server log files as described in the experiment setup. Then, 50 queries are given to five members in the Department of Computer Science. Although the queries are on quite different subjects (e.g., Information Retrieval, Machine Learning, Theory, Image Processing and Computer Network), they are all within the same domain, namely, Computer Science. Therefore, this data set is suitable for exploring the ability of adapting the search engine to a particular community of users who share similar interests but have their own specific search needs. Figure 6 provides the statistics of the data set.

Some interesting phenomena can be observed from Figure 6. On one hand, this particular group of users still bears the characteristics of the masses, such as being less likely to click on a link that is low in the ranking, no matter how relevant it

Number of queries	50	Number of clicks below rank 10	21
Total number of clicks	147	Average rank clicked on	5.39
Clicks per query	2.94	Pairs extracted	388
Lowest rank clicked on	14	Pairs per query	7.76

Fig. 6. Statistics on the data set

is (5.39 in the entry of 'Average rank clicked on'). On the other hand, there are 21 out of 147 clicks on links that are below rank 10. This means that the users who make these 21 clicks will scroll down the ranking much farther than ordinary users to observe more links. This provides more information for training in RSCF.

Note that 147 clicks means 147 sets[3] of preference pairs can be extracted by the algorithm given in [6]. Given a particular click l_a, all the preference pairs having the form $l_a <_{r'} l_b$ are generated, where l_b is a link within the same preference set corresponding to l_a. For example, l_7 is clicked by the user in Figure 1 and thus we can extract a set of preference pairs through this click, which is shown in the second column of Figure 2.

3.3 Offline Experiments

Offline experiments verified the improved performance of RSCF over RSVM algorithm. We split the feature vector ϕ into two subvectors ϕ_1 and ϕ_2, each of which is competent enough for ranking the search result.

$$\begin{aligned}\phi_1 : &(Rank(M,1), Rank(M,5), Rank(O,1), Rank(O,5), Rank(W,1),\\ &Rank(W,5), Sim\_U(q,l), Sim\_T(q,t))\\ \phi_2 : &(Rank(M,3), Rank(M,10), Rank(O,3), Rank(O,10), Rank(W,3),\\ &Rank(W,10), Sim\_C(q,a), Sim\_G(q,a))\end{aligned}$$

We use the feature mapping vector ϕ (see Example 1 and Section 3.1) to illustrate the idea. In the following ϕ is split into ϕ_1 and ϕ_2 in RSCF:

$$\phi_1(q,d) = (0,0,0,0,0,1,0,0); \qquad \phi_2(q,d) = (0,1,0,0,1,1,1,0.4)$$

Our approach is similar to *cross-validation* [9], which is desirable for small data sets. We run a 3-fold cross-validation and divide 147 sets of preference pairs into three subsets, denoted as P_A, P_B, and P_C, each with roughly the same number of preference pairs. We group the pairs in the same subset if they are extracted from the same query. First, we get the rankers from the union of P_A and P_B, and test the new ranker on P_C. Then, we get the rankers from the union of P_A and P_C and test on P_B. Finally, we get the rankers from the union of P_B and P_C and test on P_A. After all these are done, the performance of the rankers is averaged and reported.

Figure 7 shows the results of the comparison between α_R, which is trained on the whole feature vector ϕ, and α_A and α_B, which will be trained by using the feature subvectors ϕ_1 and ϕ_2, respectively. Here α_R is implemented by the original RSVM algorithm. The x-axis represents the number of preference pair sets used for training. The y-axis represents the prediction error for each ranker. As expected, ϕ_1 and ϕ_2 do not predict the users' preferences as well as ϕ, since the feature subvectors

[3] Clicking on the first link produces an empty set of preference pairs, as shown in the first column of Figure 2.

ϕ_1 and ϕ_2 are much smaller than ϕ and thus the quality of α_A and α_B would be affected. However, the advantage of using a co-training algorithm is seen after very few iterations in RSCF. We adopt the enlarged training collection of data set to establish a final ranker, denoted as α_C. Figure 8 shows the result of improvement between α_C and α_R, which is the ranker obtained from standard RSVM.

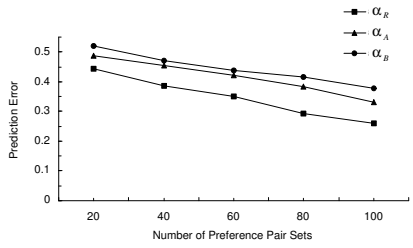

Fig. 7. The prediction errors of the rankers α_R, and the *non-trained* rankers α_A and α_B

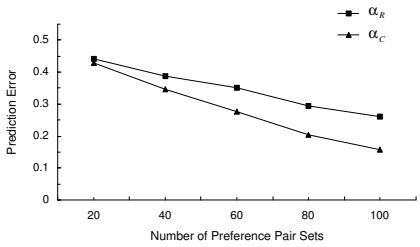

Fig. 8. The prediction errors of the rankers α_R and α_C (which combines the *trained* rankers α_A and α_B)

In order to further study the training cost in RSCF, we carry out two further tests, one on a training set of 20 queries and another one on a training set of 100 queries. While applying RSCF, at each iteration, α_A and α_B respectively selects 10 of the most confident pairs from the unlabelled data set and adds them into the labelled collection. (Recall that we enlarge P_l by choosing preference pairs from P_u as discussed in Algorithm 2.) We present our results of these experiments in Figures 9 and 10.

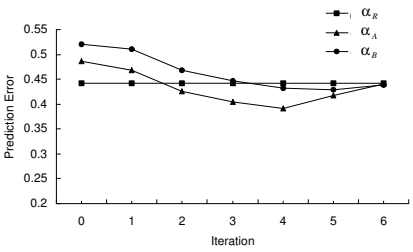

Fig. 9. The prediction errors of the rankers α_R, α_A and α_B (20 queries)

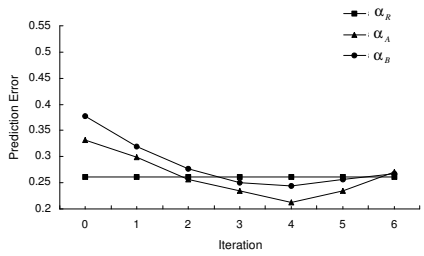

Fig. 10. The prediction errors of the rankers α_R, α_A and α_B (100 queries)

From Figures 9 and 10 we can see that, prior to the process of iteration α_R always outperforms α_A and α_B. The prediction errors of α_A and α_B decrease in the very first few iterations. Subsequently, both α_A and α_B outperform α_R at the fourth iteration, which is the local minimal in both charts. However, the prediction errors increase again after passing the local minimal. This is because at the fifth iteration and beyond, the quality of the additional top-10 preference pairs in P_u was deteriorating and selecting such pairs into P_l gives rise to a larger prediction error in the training collection. To overcome the above problem, we apply the termination criterion, *prediction difference* Δ as explained in Algorithm 2, to terminate the iteration process. The process stops when any one set of the 10 selected pairs violates $\Delta < \tau$ according to α_A and α_B. In our experiments, τ is set to 0.6.

3.4 Online Experiments

To investigate the relative performance, we evaluate the RSCF ranking function by comparing it with Google in the experiment setup outlined at the beginning of this section. For each query q, the returned ranking from RSCF, given by $r^R = (l_1^R, \ldots, l_n^R)$, and Google, given by $r^G = (l_1^G, \ldots, l_n^G)$, are combined into one shuffled list as follows, $r^{RG} = (l_1^R, l_1^G, \ldots, l_n^R, l_n^G)$ or $(l_1^G, l_1^R, \ldots, l_n^G, l_n^R)$. Note that if it happens that $l_i^R = l_j^G$ in r^{RG}, we remove the last one in the list and regard the remaining item as retrieved from both. After recording the clickthrough data, we analyze which links in r^{RG} have been clicked by the users. As it is a good indication of the relative ranking quality of the two search systems [4], we compare the number of clicks on r^R and r^G. The result is shown in Figure 11. There are 58 tested computer Science queries used in this experiment. RSCF receives more clicks for 26 queries, whereas Google receives more clicks in 17 queries. Also, there are 15 queries where users click the same number of links or none for both engines. The result shows that the proposed RSCF ranking is better in retrieval quality.

Cases	More clicks on RSCF r^R	More clicks on Google r^G	Tie	No clicks	Total
Queries	26	17	13	2	58

Fig. 11. Comparison of the learned ranking function with Google.

3.5 Experimental Analysis of Feature Vectors

We now present the analysis on the weight vector obtained from RSCF. The weights are listed in Figure 12. Intuitively, RSCF considers those features with high weights more important than those with low weights in the ranking process. That is, a document with high weights in features that have high weights will be ranked higher. In general, RSCF learnt from the clickthrough data that MSNseach is more important than Wisenut, which in turn is more important than Overture as far as ranking quality is concerned. This is consistent with the fact that Inktomi, which drives MSNsearch, has been considered one of the most mature search engines in comparison to Google, that Wisenut is a relatively new search engine and that Overture, with ranking based on the amount of money paid by advertisers, is good for product search but not good for the kind of queries used in our experiments. It is noteworthy that the binary feature $Sim\_U(q, l)$ receives more significant weight. This can be explained by the following observation. If some non-stop words in the query appear in the URL of the returned document, it is most likely that the main content in this document is related to the non-stop word. Recall that in Example

Features	**Weight**	**Features**	**Weight**
$Rank(M, 1)$	0.1914	$Rank(W, 1)$	0.0184
$Rank(M, 3)$	0.2498	$Rank(W, 3)$	0.1014
$Rank(M, 5)$	0.1152	$Rank(W, 5)$	-0.3021
$Rank(M, 10)$	0.2498	$Rank(W, 10)$	-0.4367
$Rank(O, 1)$	-0.1673	$Sim\_U(q, l)$	0.5382
$Rank(O, 3)$	-0.1229	$Sim\_T(q, t)$	0.4928
$Rank(O, 5)$	-0.4976	$Sim\_C(q, a)$	0.4136
$Rank(O, 10)$	0.4441	$Sim\_G(q, a)$	0.5010

Fig. 12. Learned weights of the features for the ranker α_C

1 the URL of d containing the word "biometrics" has good relevance in content. Another observation from Figure 12 is that the abstract has two high weight values $Sim\_C(q, a) = 0.41$ and $Sim\_G(q, a) = 0.50$, which indicates the fact that the users estimate the relevance of the document mostly based on the abstract of a document.

4 Concluding Remarks

We propose a new algorithm called Ranking SVM in a Co-training Framework (RSCF), which is able to improve the retrieval quality of search result by learning from clickthrough data. The RSCF algorithm relies only on implicit user feedback and does not add any burden to the users during the process of web searching. A key insight of utilizing clickthrough data is that the data only provides a small set of training data in the form of relative preferences. In RSCF, we resolve this problem by augmenting the training data set using a co-training framework. We show by an extensive set of offline and online experiments that RSCF improved the retrieval quality.

There are still a lot of issues that deserve further study in order to improve the performance of RSCF. In the process of feature extraction, we choose the features that we think have significant impacts on the final ranking. We plan to study other criteria for feature selection and conduct experiments to see the impacts on the performance. Another promising direction is to apply our technique to cater for individuals' needs in addition to adapting the search engine to a community of users. We believe that RSCF can be directly used to personalize search engines if the personal clickthrough data can be specifically recorded. We plan to derive ways to identify sessions of clickthrough data in log files.

References

1. Bartell, B., G.Cottrell, Belew, R.: Automatic combination of multiple ranked retrieval systemss. In: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, Dublin, Ireland (1994) 173–181
2. Cohen, W., Shapire, R., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research **10** (1999) 243–270
3. Fuhr, N.: Optimum polynomial retrieval functions based on the probability ranking principle. ACM Transactions on Information Systems **7** (1989) 183–204
4. Joachims, T.: Evaluating retrieval performance using clickthrough data. In: Proceedings of the ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval, Tampere, Finland (2002)
5. Boyan, J., Freitag, D., Joachims, T.: A machine learning architecture for optimizing web search engines. In: Proceedings of the AAAI workshop on Internet-Based Information Systems, Portland, Oregon (1996)
6. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada (2002) 133–142
7. Bennet, K., Demiriz, A.: Semi-supervised support vector machines. Advances in Neural Information Processing Systems **11** (1998) 368–374
8. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory, Madison, Wisconsin, United States (1998) 92–100
9. Goutte, C.: Note on free lunches and cross-validation. Neural Computation **9** (1997) 1245–1249