Adapting Document Ranking to Users' Preferences using Click-through Data

Min Zhao[1]*, Hang Li[2], Adwait Ratnaparkhi[3], Hsiao-Wuen Hon[2], and Jue Wang[1]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing, China.
[2] Microsoft Research Asia, Beijing, China.
[3] Microsoft Corporation, Redmond, WA, USA

Abstract. This paper proposes a new approach to ranking the documents retrieved by a search engine using click-through data. The goal is to make the final ranked list of documents accurately represent users' preferences reflected in the click-through data. Our approach combines the ranking result of a traditional IR algorithm (BM25) with that given by a machine learning algorithm (Naïve Bayes). The machine learning algorithm is trained on click-through data (queries and their associated documents), while the IR algorithm runs over the document collection. We consider several alternative strategies for combining the result of using click-through data and that of using document data. Experimental results confirm that any method of using click-through data greatly improves the preference ranking, over the method of using BM25 alone. We found that a linear combination of scores of Naïve Bayes and scores of BM25 performs the best for the task. At the same time, we found that the preference ranking methods can preserve relevance ranking, i.e., the preference ranking methods can perform as well as BM25 for relevance ranking.

1 Introduction

This paper is concerned with the problem of improving document ranking in information retrieval by using *click-through data*. Suppose that we have a search engine. Given a query, it can return a list of documents ranked based on their relevance to the query. During the use of the search engine we have collected users' click-through data. The click-through data can represent the *general trends of users' preferences* on documents with respect to queries. Our goal here is to re-rank the documents in future search using the click-through data so that the most likely to be clicked documents are re-ranked on the top from among the relevant documents. That is to say, we are to re-rank documents on the basis of users' preferences among the relevant documents.

Usually users only look at the top ranked documents in search results (cf., [22] [23] [24]), and thus if we can rank users' preferred documents on the top, then we will be able to improve users' search experiences, for example, save users' time in search.

---

[1]* Min Zhao is currently researcher at NEC Lab China, Beijing.

To the best of our knowledge, no investigation has been conducted on the problem, although there is some related work as described in Section 2.

In this paper, we employ a machine learning approach to address the above problem and investigate a number of simple, but in our view, fundamental methods.

We employ BM25 for the relevance ranking in the initial search engine (In our experiments, the search engines used for collecting click-through data were also based on BM25). We use the click-through data to train a classifier which can categorize queries into documents with scores. Specifically, queries are viewed as instances, documents are viewed as categories, and queries invoking clicks of a document are regarded as instances 'classified' into the category. As classifier, we make use of Naïve Bayes. Given a new query, BM25 and Naïve Bayes can respectively construct a ranked list of documents, on the basis of relevance and preference respectively. We consider a number of methods to combine the two lists. Linear combination is the main method among them.

Two questions may arise here. First, do the preference ranking methods preserve relevance? In other words, are the documents ranked top by the methods still relevant? Second, are the preference ranking methods indeed effective? Which method is more effective in performing the task?

We conducted experiments in order to answer the above questions, using data from two search engines: Encarta and Microsoft Word Online Help. First, experimental results indicate that our preference ranking methods can preserve *relevance ranking*. Second, experimental results indicate that the preference ranking methods using click-through data significantly perform better than BM25 for *preference ranking*. Furthermore, among the methods, the linear combination of scores from BM25 and Naïve Bayes performs the best. (We evaluated preference ranking results in terms of *click distribution*).

In conclusion, our methods can significantly improve preference ranking and at the same time preserve relevance ranking.

2 Related Work

2.1 Use of Click-through Data

Document retrieval can be described as the following problem. A search engine receives a query from a user, and then returns a ranked list of retrieved documents based on their *relevance* to the query. The list contains the links to the documents, as well as the titles and snippets of the documents. The user clicks the links of the documents which he considers *relevant and interesting*, after reading the titles and snippets.

Click-through data can be recorded in operation of the search engine, as described in [11]. Here, click-through data refers to <query, clicks> pairs recorded during document retrieval. The clicks are the set of documents clicked by the user after sending the query.

Some work has been done on using query logs in information retrieval and web data mining (e.g., [1]), because query logs contain a large amount of useful information. Here, by query logs we mean all the information which can be obtained during search of documents. Query log data, thus, contains click-through data. (Note that some researchers do not distinguish the two terms.)

Methods of using click-through data have been proposed for meta-search function training, term expansion, query clustering, implicit feedback, etc.

For example, Joachims [11] proposes a method of utilizing click-through data in learning of a meta-search function. Specifically, he employs a method called Ranking SVM. His method is unique in that it takes the relative positions of the clicks in a ranked list as training data. He has applied the method to the adaptation of a meta-search engine to a particular group of users.

Oztekin et al [17] propose several methods for meta-search. They have compared the methods by using a new metric called 'average position of clicks' calculated based on click-through data. The authors claim that the implicit evaluation method based on users' clicks can offer more statistically reliable results than an explicit evaluation method based on users' relevance judgments.

Cui et al [6] try to use click-through data to solve the problem of mismatch between query terms and document terms. With click-through data, the probabilistic correlations between query terms and documents terms can be extracted and high quality term expansion can be conducted.

Furthermore, Beeferman etc. [4] propose methods for harvesting clusters of queries and web pages by using click-through data. Ling et al [14] investigate a similar problem, but focus on the issue of finding generalized query patterns and using them in the cache of a search engine.

In [12] [18] and [21], click-through data is regarded as implicit feedback and used to improve relevance ranking.

DirectHit was a commercial internet search engine (http://www.directhit.com). It is claimed that DirectHit uses click-through data in ranking of retrieved documents (http://www.searchengines.com/directhit.html). Their problem setting is similar to that in the current research; however, details of the technologies used are not known.

2.2 Meta Search and Relevance Feedback

Meta-search attempts to improve ranking by combining ranking results returned by several search engines, while relevance feedback manages to improve ranking by using feedbacks on relevance from users. Both of them consider performing ranking based on *relevance*, not based on users' *preference*.

Meta-search and related problems such as result fusion have been intensively studied [2] [3] [8] [9] [11] [13] [15] [17] [25]. One of the key issues in meta-search is to construct a meta-ranking function which combines the scores or ranks returned by ranking functions. Many meta-ranking functions have been proposed (cf., [2] [16]), which are based on averaging, sum, linear combination, voting, interleaves, logistic regression, etc.

Relevance feedback is an iterative process as follows. Given documents returned by the search engine, the user is asked to mark them as relevant or irrelevant. The query is then expanded with terms extracted from the marked relevant documents. Next, documents are retrieved with the new query. The process is repeated (cf., [20]). Since relevance judgments can be burdens to users, automatic ways of getting feedbacks, such as pseudo-relevance feedback [5] and implicit relevance feedback [21] [26], have been proposed. The main purpose of relevance feedback is to refine user's query so that user's information needs can be expressed more accurately.

3 Problem and Evaluation

3.1 Preference Ranking Using Click-through Data

The problem we address in this paper is whether and how we can use click-through data to improve preference ranking. More specifically, we consider a re-ranking (or adaptation) problem as follows: a search engine first receives a query from a user, and then presents a ranked list of retrieved documents based on their relevance to the query. (We assume here that the document collection and the relevance ranking mechanism (e.g., BM25) are fixed). Next, a re-ranking engine re-orders the ranked list of documents based on users' preferences, using click-through data. The top ranked documents should be relevant and preferred by users.

In one extreme case, the re-ranking engine completely ignores the initial relevance ranking and creates a new preference ranking. In the other extreme case, the re-ranking engine only uses the initial relevance ranking.

Preference and relevance are different notions, although closely related to each other. If users think that some documents are more worth-reading (important and interesting) than the others among the relevant documents, then we say that they prefer those documents. In this paper, preferences are assumed to be from a group of users, not a single user. Preference is more subjective and dynamic, while relevance is more objective and static. When a query is fixed, its relevant documents are almost fixed, while users' preferences may change depending on user groups and time periods.

In this paper, we assume that click-through data can reflect the general trends of users' preferences. (Note that users' preferences can be measured by other factors like 'dwell time', cf., [7]). It is likely that a user clicks a document which he find uninteresting or even irrelevant later. The percentage of such 'noisy' clicks over total clicks should be low, however. (Otherwise, the search engine will not be used by users.) Therefore, click-through data can and should be used in preference ranking. We also assume here that there is no 'click spam'.

Figure 1 shows the differences between the current problem and other information retrieval problems: conventional search, meta-search, relevance feedback. The current problem focuses on preference ranking, while the other problems focus on relevance ranking.

In practice, usually users only look at the top ranked documents in search results (cf., [22] [23] [24]), and thus putting users' preferred documents on the top is necessary and important for document retrieval.

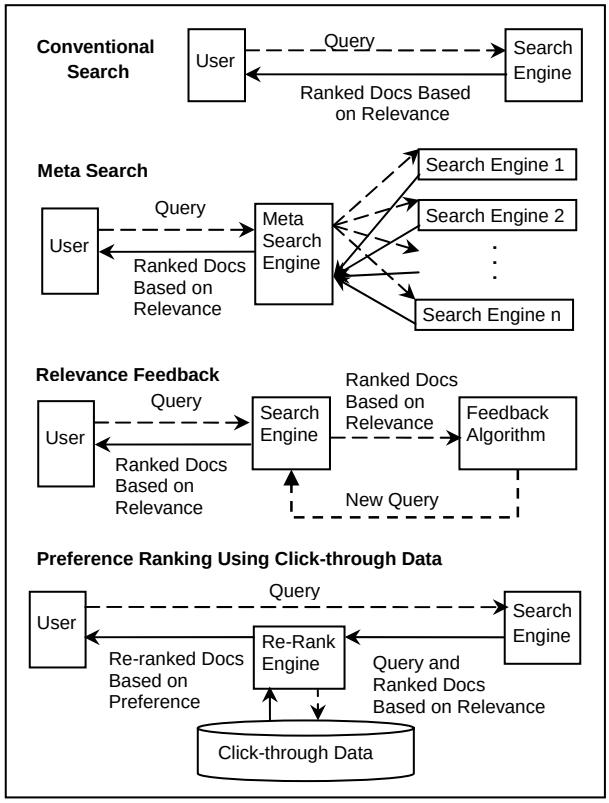

Fig. 1. Comparison of retrieval problems

3.2 Click Distribution

An ideal way of evaluating the performance of a preference ranking method might be to select a number of users and a set of queries, and let every user to explicitly judge preference on each of the documents to each of the queries. Furthermore, for each query, counting the number of preference judgments on each document, ranking the documents in descending order of their preference counts, and taking the order as the correct 'answer'. This evaluation can give accurate results. However, it is costly, and thus is not practical.

We consider here the use of '*click distribution*' for the evaluation. Click distribution is the distribution of document positions clicked by users in the ranked lists. If a user clicks *k* documents, a method that places the *k* documents in higher ranks (ideally the first k ranks) is better than a method that does not.

The advantage of using the measure is that a large amount of click-through data can be used. The disadvantage is that click-through data may contain noise. However, as is explained above, the percentage of noise should be low. Thus, click distribution is a good enough measure for evaluation of preference ranking methods.

4 Methods

We propose a machine learning approach to address the problem of preference ranking using click-through data. It turns out to be a problem of using click-through data to re-rank documents so that the most likely to be clicked documents are re-ranked on the top from among the relevant documents.

We first use click-through data to train a classifier which categorizes queries into documents with scores. It is actually a problem of text classification. Specifically, we view queries as instances, documents as categories, and queries invoking clicks of a document as instances 'classified' into the category. We next combine the results given by the classifier and the results given by the initial search engine. It is actually a problem of result fusion. Specifically, we combine the information from two data sources: the classifier and the search engine, in order to achieve better performances than using information from either of the sources alone.

As the initial search engine, we employ the ranking method used in the Okapi system [19], which is called BM25 (*BM* hereafter). (This is because in the experiments described below click-through data were collected from search engines based on BM25.)

For click-through data classification, we employ Naïve Bayes (hereafter *NB*).

For combination of the two methods, we consider a number of strategies: (1) data combination (*DC*), (2) ranking with NB first and then BM (*NB+BM*), and (3) linear combination of BM and NB scores (*LC-S*) and linear combination of BM and NB ranks (*LC-R*). BM and NB themselves are the combination methods in two extreme cases: the former does not make use of click-through data and the latter makes only use of click-through data. DC first combines click-through data with document data, and then uses BM.

In practice, some search engines only return a ranked list of documents without scores. NB, NB+BM, and LC-R can still work under such circumstances.

4.1 Notation

$D = \{d\}$ is a document collection.

$Q = \{q\}$ is a set of queries, and a query q consists of words.

$T = \{t\}$ is a set of words.

$R(q) = <d_1, d_2, \cdots d_m>$ is a ranked list of retrieved documents with respect to query q, returned by a method. Some methods only return a ranked list, while other methods associate each d_i ($1 \leqslant i \leqslant m$) with a score $S(q, d)$, satisfying $S(q, d_1) \geqslant S(q, d_2) \geqslant \cdots \geqslant S(q, d_m)$.

$C = \{<q, c>\}$ is a set of click-through data, where $c = <d_1, d_2, \cdots d_k>$ is a set of documents clicked by a user.

$Qd = \{q\}$ is the subset of Q related to document d in click-through data.

4.2 BM – Initial Ranking Function

In BM, given query q, we first calculate the following score (using the default values of parameters in the formula of BM25) of each document d with respect to it

$$S_{BM}(q,d) = \sum_{t \in q} \left\{ \frac{tf_{t,d} \cdot \log(\frac{N - df_t + 0.5}{df_t + 0.5})}{0.5 + 1.5\frac{dl_d}{dl_{avg}} + tf_{t,d}} \right\},$$

where $tf_{t,d}$ is term frequency of word t in document d, df_t is document frequency of word t in collection D, N is number of documents in D, dl_d is length of document d, and dl_{avg} is average length of documents in D.

We next rank documents according to their scores $S_{BM}(q,d)$, and then obtain a ranked list of documents $R_{BM}(q)$.

4.3 NB – Click-through Classification

We can view each document in the given collection as a category, and the queries associated with each of the documents as instances classified into the category. The association can be found in click-through data (cf., Figure 2). Thus, we can construct a classifier on the basis of the click-through data and formalize the problem of document retrieval as that of text classification.

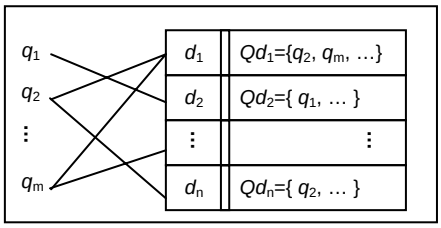

Fig. 2. Relationship between queries and documents

In NB, we employ Naïve Bayes as classifier [16]. Given query q (a sequence of words $t_1, t_2, \ldots, t_l$) and category d, the conditional probability of $p(d|q)$ is calculated as follows.

$$p(d \mid q) = \frac{p(d) \cdot p(t_1, t_2, \ldots, t_l \mid d)}{p(t_1, t_2, \ldots, t_l)} \propto p(d) \cdot \prod_{t \in d} p(t \mid d) \cdot$$

We calculate the probabilities using click-through data.

$$p(d)\cdot\prod_{t\in q}p(t\mid d)=p(d)\cdot\prod_{t\in q}\left\{\frac{tf_{t,Qd}+1}{tf_{Qd}+|V_Q|}\right\}.$$

Similarly we have

$$p(\bar{d}\mid q)=p(\bar{d})\cdot\prod_{t\in q}p(t\mid\bar{d})=p(\bar{d})\cdot\prod_{t\in q}\left\{\frac{tf_t-tf_{t,Qd}+1}{tf_Q-tf_{Qd}+|V_Q|}\right\}.$$

Here, Qd is subset of queries in Q associated with document d, $tf_{t,Qd}$ is term frequency of word t in Qd, tf_t is term frequency of t in Q, tf_{Qd} is total term frequency in Qd, tf_Q is total term frequency in Q, and $|V_Q|$ is number of unique words in Q. Furthermore, $p(d)$ and $p(\bar{d})$ are prior probabilities of category d and its converse category $\bar{d}$, and we assume that they are equal for all documents. The converse category $\bar{d}$ consists of all documents except d.

We use log of ratio of the two probabilities as score of document d with respect to query q.

$$S_{NB}(q,d)=\log\left\{\frac{p(d\mid q)}{p(\bar{d}\mid q)}\right\}.$$

In NB, we rank documents by $S_{NB}(q,d)$ and then obtain $R_{NB}(q)$.

4.4 DC – Data Combination

The above two methods rank documents based upon information from only one data source: either document or click-through. In the sequel, we consider a number of methods which combine the information from both data sources.

The simplest method of combination might be to combine the queries to their associated documents, view them as pseudo-documents, and run BM with the pseudo-documents.

More precisely, for each document d and its associated query set Qd, we create a pseudo-document d' by combing d and Qd. Given a new query q, apply BM to the collection $D'=\{d'\}$ and get $S_{BM}(q,d')$ for each document. Let $S_{DC}(q,d) = S_{BM}(q,d')$, we can rank the documents with the scores and obtain $R_{DC}(q)$.

4.5 NB+BM – NB First and BM Next

In NB+BM, given the initial ranked list by BM, we pick up from the list those documents whose NB scores are larger than a predetermined threshold, rank them by NB, and put them on the top of the list. The assumption is that click-through data reflects users' preferences and thus it is better to use the information first for preference ranking.

Let $R_{BM}(q)$ be the ranked list of documents returned by BM. Some of them can be ranked by NB with the ranked list denoted as $R_{NB}(q)$. We put the documents in $R_{NB}(q)$ at the top, and the remaining documents behind according to $R_{BM}(q)$. We then obtain $R_{NB+BM}(q)$. For example, if $R_{BM}(q) = <d_1, d_2, d_3, d_4>$ and $R_{NB}(q) = <d_3, d_2>$, then $R_{NB+BM}(q) = < d_3, d_2, d_1, d_4>$.

4.6 LC-S and LC-R – Linear Combination

We can employ a fusion strategy to combine the results of NB and BM. The most widely used method for such fusion is linear combination.

We consider two sorts of linear combination methods. In *Linear Combination of Scores* (LC-S), we rank documents by linearly combining the normalized scores returned by BM and NB:

$$S_{LC-S}(q,d) = S_{BM}(q,d) + \alpha \cdot S_{NB}(q,d) \ ,$$

where $S_{BM}(q,d)$ and $S_{NB}(q,d)$ are normalized to [0, 1], α is the weight in (0, 1).

In *Linear Combination of Ranks* (LC-R) we rank documents by linearly combining the normalized ranks returned by BM and NB. Suppose that document *d* is ranked at i^{th} position in $R_{BM}(q)$ and at j^{th} position in $R_{NB}(q)$, and there are *m* documents in $R_{BM}(q)$ and *n* documents in $R_{NB}(q)$, then we have

$$S_{LC-R}(q,d) = \frac{m-i}{m} + \alpha \cdot \frac{n-j}{n} \ ,$$

where α is weight.

We determine the weight α in the linear combination by training on click-through data.

As measure for training, we use *Percentage of Clicks on Top N* (*PC-TopN* for short). PC-TopN represents the percentage of clicked documents on top-N positions among all clicked documents. Actually, PC-TopN is the result at point *N* in a *click distribution*. Since we use click distribution for evaluation, it is natural to use the same measure for training (For training, we can use a single value instead of a distribution as the measure).

Given a specific value of α, we can create a linear combination model and calculate its corresponding PC-TopN value with the *training* data. From a number of possible values of α, we select the one that maximizes PC-TopN with respect to the training data. That is to say, we choose the α that achieves the best performance in document preference ranking. Since there is only one parameter α in our linear combination model, there is no need to employ a complicated parameter optimization method.

5 Experiments and Results

We conducted two experiments on two data sets. One was to investigate whether our preference ranking methods can preserve relevance ranking. The other was to examine whether our methods outperform the baseline BM25 for preference ranking, and among them which method works best.

5.1 Data Sets

The two data sets used in our experiments are 'Encarta' and 'WordHelp'. Encarta is an electronic encyclopedia on the web (http://encarta.msn.com), which contains 44,723 documents. Its search engine (based on BM) has recorded click-through data. We obtained data collected over two months. WordHelp is from Microsoft Word Online Help, which contains 1,340 documents. In an experimental setting, its search engine (also based on BM) collected click-through data during four months. Details of the two data sets are shown in Table 1.

Table 2 shows randomly selected queries in the click-through data of each data set.

The titles and snippets in both WordHelp and Encarta are created by humans, and thus are very accurate. Thus, for the two data sets it is possible for users to make preference judgments based on title and snippet information.

Table 1. Details of WordHelp and Encarta click-through data

	Word	Encarta
Time span	4 months	2 months
Total number of queries	~34,000	~4,600,000
Total number of unique queries	~13,000	~780,000
Average query length	~1.9	~1.8
Average clicks per query	~1.4	~1.2

Table 2. Example queries in WordHelp and Encarta click-through data

WordHelp	xml, table contents, watermark, signature, speech, password, template, shortcut bar, fax, office assistant, equation editor, office shortcut bar, highlight, language, bookmark
Encarta	electoral college, China, Egypt, Encarta, sex, George Washington, Christmas, Vietnam war, dogs, world war II, solar system, abortion, Europe

5.2 Training and Testing Data

The click-through data in Encarta and WordHelp were recorded in chronological order. We separated them into two parts according to time periods, and used the first parts for training and the second parts for testing.

For Encarta, we had 4,500,000 <*q*, *c*> click-through pairs for training, and 10,000 pairs for testing. For WordHelp, we had 30,000 <*q*, *c*> click-through pairs for training and 2,000 pairs for testing.

For NB and NB+BM, we made use of the training data (click-through data) in the construction of classifiers. For LC-R and LC-S, we made use of the training data in

the construction of classifiers and a randomly selected subset of training data (10,000 instances in Encarta and 2,000 instances in WordHelp) in the tuning of linear combination weights. For DC, we utilized the training data and the document data in the construction of models.

When training the weights in LC-R and LC-S, we tried different values of N for PC-TopN: PC-Top1, PC-Top2, PC-Top5, PC-Top10, and PC-Top20. It seems that N only slightly affects the weights and the final ranking results have little differences with different values of N used in training. Thus, in the experiments reported in this paper, N was fixed at 5.

In order to conduct precise analysis, we further created three data sets on the basis of the same click-through data for both Encarta and WordHelp. They are referred to as one_click, first_click, and all_clicks, respectively. In one_click, we used the $<q, c>$ pairs only having single clicks as instances. In first_click, we extracted the first clicks in all click-through pairs to create instances. In all_clicks, we used all click-through pairs as instances. For example, given two click-through instances $<q_1, \{d_1, d_2\}>$ and $<q_2, \{d_3\}>$, one_click = $\{<q_2, d_3>\}$, first_click = $\{<q_1, d_1>, <q_2, d_3>\}$, and all_clicks = $\{<q_1, d_1, d_2>, <q_2, d_3>\}$. Due to limitation of space, we only report the results with first_click in this paper. The results with one_click and all_clicks have the same tendencies.

5.3 Experiment 1

We randomly selected 40 queries from the testing (click-through) data of Encarta and that of WordHelp respectively. More precisely, for each data set, we sorted all the queries in terms of frequency, then equally divided the sorted set into four subsets, and randomly picked up 10 queries from each of the subsets. In this way, we were able to select queries from different frequency ranges.

For each query, the six ranking methods described in Section 4 were employed to produce a ranked list, and the top-5 documents returned by the methods were merged together. Next, humans made judgments on the relevance of the documents. We then used the measure of top 1, 2, and 5 precisions to evaluate the performances of the six methods for *relevance ranking*.

We report here the results when 10% of Encarta and 20% of WordHelp training data were respectively used for training of the ranking methods, in accordance with the results reported for Experiment 2 in Section 5.4.

The results in Table 3 indicate that all the *preference* ranking methods using click-through data improve upon or at least work as well as the initial relevance ranking method – BM. (Some of the improvements are statistically significant according to our statistical testing). Thus, it is safe to say that the preference ranking methods can preserve relevance ranking.

It should be noted that NB, the method which ranks documents by using click-through data alone, can still achieve better results than BM. The result indicates that preference is strongly related to relevance.

Table 3. Performance of six methods with respect to WordHelp and Encarta data in terms of top 1, 2 and 5 precisions (Pre@1, Pre@2, Pre@5)

Collection	Method	Pre@1	Pre@2	Pre@5
WordHelp	LC-S	**0.850**	**0.700**	**0.455**
	NB+BM	0.700	0.625	0.450
	NB	0.700	0.613	0.445
	DC	0.700	0.600	0.435
	LC-R	0.675	0.575	0.390
	BM	0.650	0.537	0.390
Encarta	LC-S	**0.825**	**0.563**	**0.300**
	NB+BM	0.750	0.525	0.280
	NB	0.750	0.525	0.280
	DC	0.750	0.525	0.270
	LC-R	0.725	0.500	0.260
	BM	0.450	0.388	0.250

5.4 Experiment 2

We applied the methods of BM, NB, DC, NB+BM, LC-R, and LC-S to both Encarta and WordHelp data for preference ranking.

All the results with Encarta and WordHelp show almost the same tendencies.

We used different proportions of training data in creating NB, DC, NB+BM, LC-R, and LC-R. We only show the results here when 20% and 100% of WordHelp training data, and 10% and 100% of Encarta training data are available. This is because they are representative of the general trends in the entire results.

We found that the use of click-through data is effective for enhancing preference ranking. Nearly in all cases, the methods using click-through (i.e., LC-S, LC-R, NB, NB+BM, DC) perform better than the method without using it (i.e., BM).

We also found that when the size of training (click-through) data is large, LC-S, NB+BM, and NB perform nearly equally best, and when the size of training (click-through) data is small, LC-S performs best. Therefore, it is better to always employ LC-S.

The results were obtained on the basis of the measure: click distribution as described in Section 3.2. Again, click distribution represents the percentage of clicked documents on top N positions among clicked documents. The higher percentage of clicks on top, the better a method is. For each method, we evaluated the percentages of clicks on top 1, top 2, top 10, top 20, and all. We also evaluated the percentages of clicks in intervals between rank 1-1, rank 2-2, rank 3-5, rank 6-10, rank 11-20, and rank 21-.

Figure 3 shows the click distributions when 10% of Encarta and 20% of WordHelp training (click-through) data are respectively available. Among the methods, LC-S performs the best.

Figure 4 shows the click distributions when 100% training data is available. Among the methods, LC-S, NB+BM, and BM perform best.

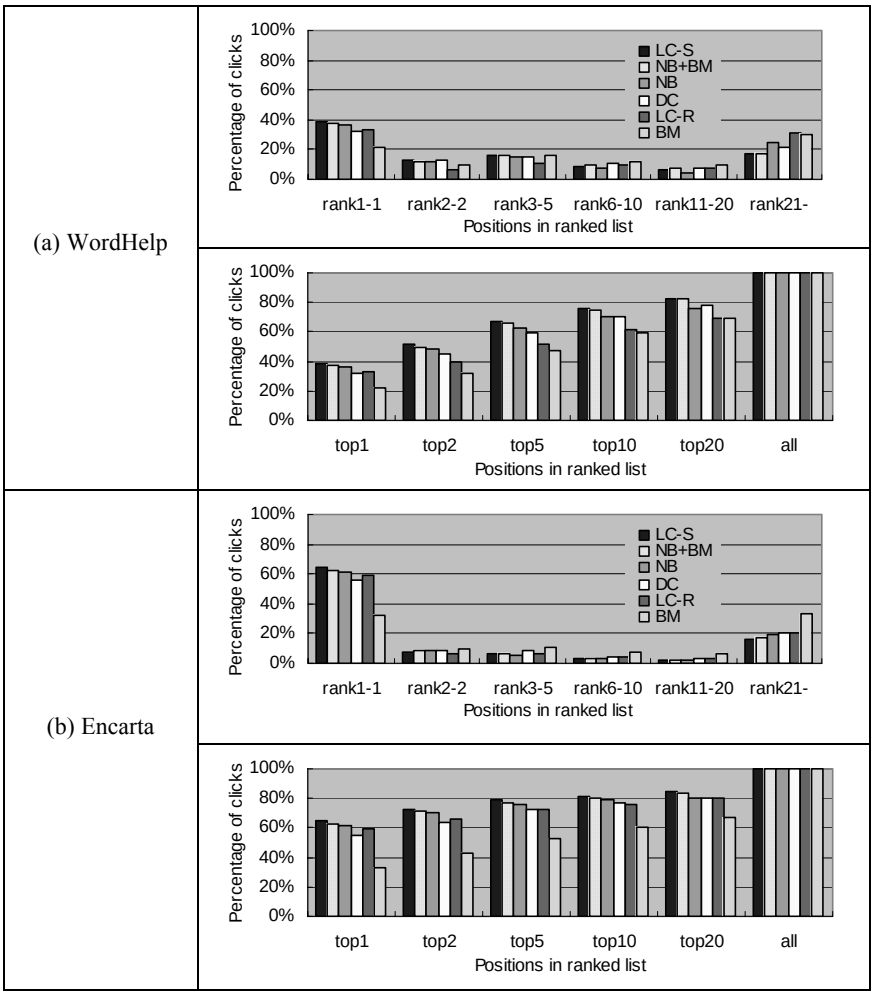

Fig. 3. Click distributions with 20% WordHelp training data and 10% Encarta training data

We conducted sign test [10] to see the significance of differences between methods on percentage of clicks on top5 (significance level was set to 0.005).

We use '{A, B}' to represent 'methods A and B do not have significant difference in performance', and use 'A >> B' to represent 'method A is significantly better than method B'.

With 20% training data for WordHelp and 10% training data for Encarta, we have:

(1) WordHelp: {LC-S, NB+BM} >> NB >> DC >> LC-R >> BM

(2) Encarta: LC-S >> NB+BM >> NB >> {DC, LC-R} >> BM

With 100% training data, we get the same results for WordHelp and Encarta:

(3) {LC-S, NB+BM, NB} >> {DC, LC-R} >> BM.

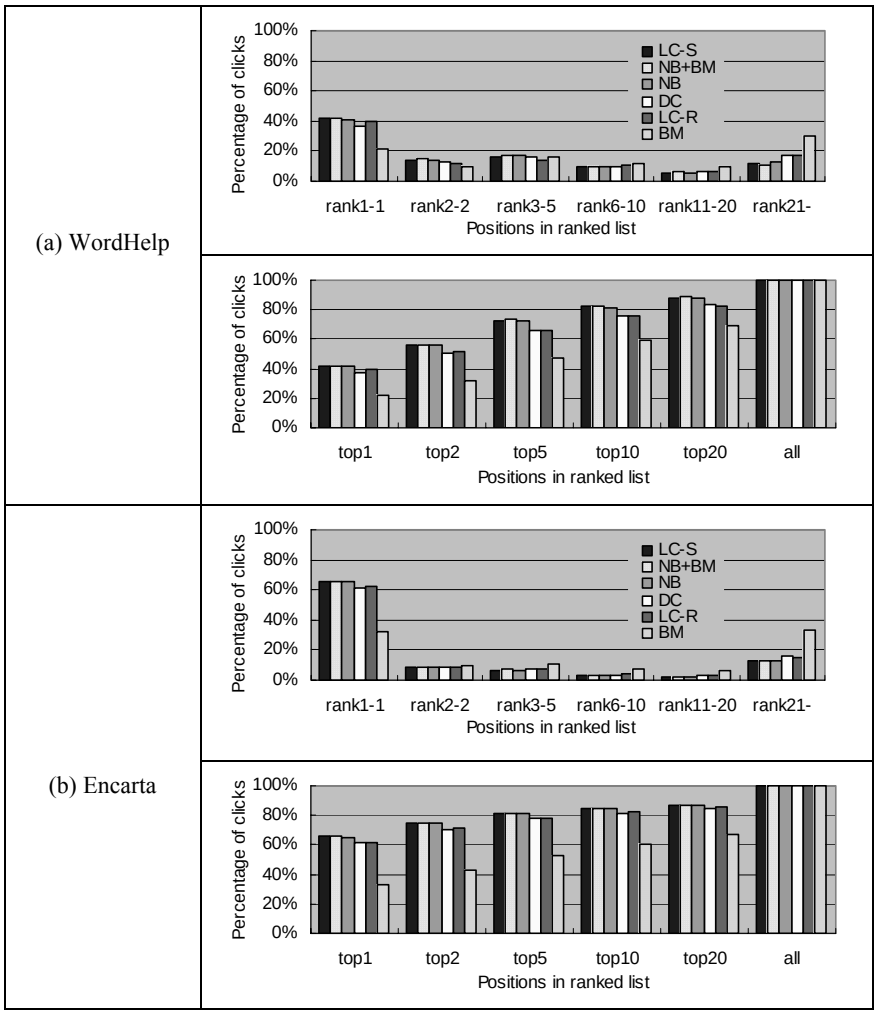

Fig. 4. Click distributions with 100% WordHelp and Encarta training data

5.5 Discussions

We discuss why the use of click-through data can help improve preference ranking and why LC-S can work best when different sizes of click through data are available. We examined how the proportion of 'unseen' queries in testing data changes when training (click-through) data gets accumulated. Figure 5 shows the results. We see that the number of previously unseen queries in the testing data decreases when training data increases no matter whether it is in time order or reverse order. For WordHelp, eventually only 34% of testing queries is previously unseen, and for Encarta, only 15% of testing queries is previously unseen. Therefore, using click-

through data (already observed click records) can help improve prediction of users' future clicks, i.e., preference ranking.

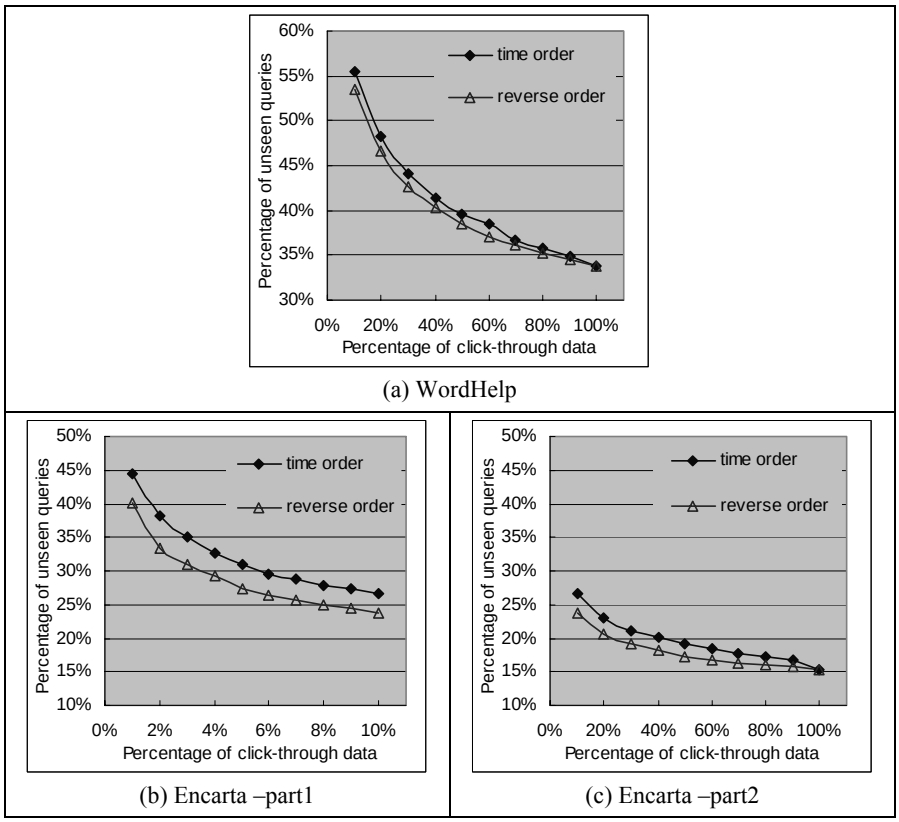

Fig. 5. Percentages of unseen queries in testing data versus sizes of WordHelp and Encarta training data

Table 4. Distribution of queries and clicked documents in click-through data (Type represents unique queries / clicked documents, and Token represents all queries / clicked documents)

Frequency		1	2-3	4-10	11-50	51-
WordHelp query	Type	76.5%	14.1%	6.3%	2.6%	0.5%
	Token	29.9%	12.5%	14.4%	21.5%	21.7%
WordHelp clicked document	Type	3.2%	5.7%	22.6%	49.5%	19.0%
	Token	0.1%	0.4%	4.6%	35.8%	59.1%
Encarta query	Type	68.3%	19.7%	7.4%	3.1%	1.5%
	Token	11.4%	7.4%	7.2%	11.9%	62.1%
Encarta clicked document	Type	8.0%	11.5%	20.4%	29.1%	31.0%
	Token	0.1%	0.2%	0.9%	4.8%	94.0%

Table 4 shows the frequency distributions of queries and clicked documents in the click-through data. For both data sets, query frequency is heavily distributed on a small number of common terms, and clicked document frequency is heavily distributed on a small number of common documents. The statistics indicate that

prediction of users' clicks based on click-through data is at least possible for common queries.

From the results, we see that the way of using click-through data, as we propose, can effectively enhance the performance of preference ranking.

It appears reasonable that NB, NB+BM, LC-S perform equally well when click through data size is large enough. In such situation, NB performs much better than BM, and thus both in the 'back-off' model of NB+BM and the linear combination model of LC-S, NB does dominate (i.e., NB either works frequently or has a large weight).

Interestingly, when click-through data is not sufficient, LC-S performs best, suggesting that combining information from two different sources effectively helps improve performance even if training data is not enough.

It is easy to understand LC-S always performs better than LC-R, as the former uses scores and the latter uses ranks, and there is loss in information when using LC-R.

6 Conclusions

We have investigated the problem of preference ranking based on click-through data, and have proposed a machine learning approach which combines information from click-through data and document data. Our experimental results indicate that

- click-through data is useful for preference ranking,
- our preference ranking methods can perform as well as BM25 for relevance ranking,
- it is better to employ a linear combination of Naïve Bayes and BM25.

Future work still remains. In our study, we used Naïve Bayes as the classifier. It will be interesting to see how other classifiers can work on the problem. In our experiments, we tested the cases in which the document collection was fixed. It is an open question whether we can make a similar conclusion in the cases in which the document collection dynamically changes during recording of click-through data.

References

1. Anick, P. Using terminological feedback for web search refinement – a log-based study. In Proceedings of SIGIR'03 (2003) 88-95.
2. Aslam, J.A., and Montague, M.H. Models for Metasearch. In Proceedings of SIGIR'01 (2001) 275-284.
3. Bartell, B.T., Cottrell, G.W., and Belew, R.K. Automatic combination of multiple ranked retrieval systems. In Proceedings of SIGIR'94 (1994) 173-181.
4. Beeferman, D., and Berger, A. Agglomerative clustering of a search engine query log. In Proceedings of SIGKDD'00 (2000) 407-416.
5. Buckley, C., Salton, G., Allan J., and Singhal A. Automatic Query Expansion Using SMART: TREC 3. TREC-1994 (1994) 69-80.

6. Cui, H., Wen, J.R., Nie, J.Y., and Ma, W.Y. Probabilistic query expansion using query logs. In Proceedings of WWW'02 (2002) 325-332.
7. Dumais, S., Joachims, T., Bharat, K, and Weigend, A. SIGIR 2003 Workshop Report: Implicit Measures of User Interests and Preferences. SIGIR Forum, 37(2), Fall 2003. (2003) 50-54.
8. Fox, E.A. and Shaw, J.A. Combination of multiple searches. In Proceedings of TREC-2 (1994) 243-249.
9. Greengrass, E. Information Retrieval: a Survey (2000). http://www.cs.umbc.edu/cadip/readings/IR.report.120600.book.pdf
10. Hull, D.A. Using Statistical Testing in the Evaluation of Retrieval Experiments. In Proceedings of SIGIR'93 (1993) 329-338.
11. Joachims, T. Optimizing search engines using clickthrough data. In Proceedings of SIGKDD'02 (2002) 133-142.
12. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G. Accurately interpreting clickthrough data as implicit feedback. SIGIR'2005 (2005) 154-161
13. Lee, J.H. Analyses of multiple evidence combination. In Proceedings of SIGIR'97 (1997) 267-276.
14. Ling, C.X., Gao, J.F., Zhang, H.J., Qian, W.N., and Zhang, H.J. Improving Encarta search engine performance by mining user logs. International Journal of Pattern Recognition and Artificial Intelligence. 16(8) (2002) 1101-1116.
15. Manmatha, R., Rath, T., and Feng, F. Modeling score distributions for combining the outputs of search engines. In Proceedings of SIGIR'01 (2001) 267-275.
16. Mitchell, T.M. Machine Learning. McGraw Hill (1997).
17. Oztekin, B., Karypis, G., and Kumar, V. Expert agreement and content based reranking in a meta search environment using mearf. In Proceedings of WWW'02 (2002) 333-344.
18. Radlinski, F., Joachims, T. Query chains: learning to rank from implicit feedback. KDD'2005 (2005) 239-248
19. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M. and Gatford, M., Okapi at TREC-3. In Overview of the Third Text REtrieval Conference (TREC-3) (1995) 109-126.
20. Salton, G., and Buckley, C. Improving retrieval performance by relevance feedback. Journal of American Society for Information Sciences. (1990) 41:288--297.
21. Shen, X., Tan, B., Zhai, C. Context-sensitive information retrieval using implicit feedback. SIGIR'2005 (2005) 43-50.
22. Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. Analysis of a Very Large AltaVista Query Log. Technical Report SRC 1998-014, Digital Systems Research Center (1998).
23. Spink, A., Jansen B.J., Wolfram, D., and Saracevic, T. From e-sex to e-commerce: web search changes. IEEE Computer, 35(3) (2002) 107-109.
24. Spink, A., Wolfram, D., Jansen B.J., and Saracevic, T. Searching the web: the public and their queries. Journal of the American Society of Information Science and Technology, 52(3) (2001) 226-234.
25. Vogt, C.C., and Cottrell, G.W. Predicting the performance of linearly combined IR systems. In Proceedings of SIGIR'98 (1998) 190-196.
26. White, R.W., Ruthven, I., and Jose J.M. The use of implicit evidence for relevance feedback in web retrieval. In Proceedings of ECIR'02 (2002) 93-109.