LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval

Tie-Yan Liu[1], Jun Xu[1], Tao Qin[2], Wenying Xiong[3], and Hang Li[1]

[1] Microsoft Research Asia, No.49 Zhichun Road, Haidian District, Beijing China, 100080

[2] Dept. of Electronic Engineering, Tsinghua University, Beijing, China, 100084

[3] Dept. of Computer Science, Peking University, Beijing, China, 100871

[1]{tyliu, junxu, hangli}@microsoft.com; [2]qinshitao99@mails.thu.edu.cn; [3]flykitej@gmail.com

ABSTRACT

This paper is concerned with learning to rank for information retrieval (IR). Ranking is the central problem for information retrieval, and employing machine learning techniques to learn the ranking function is viewed as a promising approach to IR. Unfortunately, there was no benchmark dataset that could be used in comparison of existing learning algorithms and in evaluation of newly proposed algorithms, which stood in the way of the related research. To deal with the problem, we have constructed a benchmark dataset referred to as LETOR and distributed it to the research communities. Specifically we have derived the LETOR data from the existing data sets widely used in IR, namely, OHSUMED and TREC data. The two collections contain queries, the contents of the retrieved documents, and human judgments on the relevance of the documents with respect to the queries. We have extracted features from the datasets, including both conventional features, such as term frequency, inverse document frequency, BM25, and language models for IR, and features proposed recently at SIGIR, such as HostRank, feature propagation, and topical PageRank. We have then packaged LETOR with the extracted features, queries, and relevance judgments. We have also provided the results of several state-of-the-arts learning to rank algorithms on the data. This paper describes in details about LETOR.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]

General Terms

Algorithms, Performance, Experimentations

Keywords

Learning to Rank, Benchmark Datasets

1. INTRODUCTION

Ranking is the central problem for many information retrieval (IR) applications. These include document retrieval [3], collaborative filtering [14], key term extraction [8], definition finding [32], important email routing [4], sentiment analysis [26], product rating [9], and anti web spam [13]. In the task, given a set of objects, we utilize a ranking model (function) to calculate the score of each object and sort the objects with the scores. The scores may represent the degrees of relevance, preference, or importance, depending on applications.

Learning to rank is aimed at automatically creating the ranking model using training data and machine learning techniques. A typical setting in learning to rank is that feature vectors and ranks (ordered categories) are given as training data. A ranking model is learned based on the training data and then applied to the unseen test data. Because of the advantages it offers, learning to rank has been gained increasing attention in IR. Many methods have been proposed (e.g., [2], [5], [6], [7], [10], [16], [20], [22], [23]) and applied to IR applications (e.g., [3], [12], [27], [31], [32], [33]). Unfortunately, the experimental results in the papers were obtained from different datasets and under different settings, and thus it was impossible to make a direct comparison between the results. This became a blocking issue for future research and development, in contrast to the situations in many other fields in machine learning in which research has been significantly enhanced by the availabilities of benchmark datasets, such as Reuters 21578 [36] and RCV1 [10] for text categorization, and SANBI EST [37] for clustering. Therefore, it would be very beneficial, if benchmark datasets for learning to rank could be developed. This is exactly the problem we want to address here.

In this work, we have built a benchmark dataset named LETOR. We have extracted features from two widely-used data collections in information retrieval (the OHSUMED collection used in the information filtering task of TREC 2000, and the ".gov" collection used in the topic distillation task of TREC 2003 and 2004). These two collections contain queries, their associated documents, and human judgments on the relevance of the documents with respect to the queries. The features we extracted include both conventional features (term frequency, inverse document frequency, document length, BM25 [29], and Language Models for Information Retrieval [35]) and new features proposed recently at SIGIR (HostRank [34], feature propagation [28] and topical PageRank [25]). We have packaged the extracted features, queries, and relevance judgments into the LETOR dataset. Given the fact that the topic distillation tasks of TREC 2003 and TREC 2004 have different query sets, we actually have three subsets in the LETOR dataset: OHSUMED, TD2003, and TD2004. We have partitioned each subset into a training set, a validation set, and a test set. We have run several state-of-the-arts learning to rank algorithms on the subsets and report their ranking performances in this paper. We also discuss future research directions that people can explore using the LETOR dataset.

This paper provides the details on the feature extraction, data partitioning, and baseline experimental results regarding to the LETOR dataset. We begin in Section 2 by describing the original collections of OHSUMED and ".gov". Section 3 shows the list of features we extracted from the collections, and discusses how to partition the dataset for training, validation, and testing. Section 4 reports the ranking performances of the state-of-the-arts ranking algorithms on the dataset. Section 5 discusses the potential research directions. The last section concludes the paper.

2. DATA COLLECTIONS

2.1 The OHSUMED collection

The OHSUMED collection [15] was created for information retrieval research. It is a subset of MEDLINE, a database on medical publications. The collection consists of 348,566 records (out of over 7 million) from 270 medical journals during the period of 1987-1991. The fields of a record include title, abstract, MeSH indexing terms, author, source, and publication type.

There are 106 queries, each with a number of associated documents. A query is about a medical search need, and thus is also associated with patient information and topic information. The relevance degrees of documents with respect to the queries are judged by humans, on three levels: *definitely relevant*, *partially relevant*, or *not relevant*. There are a total of 16,140 query-document pairs with relevance judgments.

The MEDLINE documents have the same file format as those in the SMART system, with each field defined as below (NLM designator in parentheses):

• *.I*	Sequential identifier
• *.U*	MEDLINE identifier (UI)
• *.M*	Human-assigned MeSH terms (MH)
• *.T*	Title (TI)
• *.P*	Publication type (PT)
• *.W*	Abstract (AB)
• *.A*	Author (AU)
• *.S*	Source (SO)

For each query in the OHSUMED collection, the patient and topic information are defined in the following way:

• *.I*	Sequential identifier
• *.B*	Patient information
• *.W*	Information request

Many research papers [33] [27] have been published using the OHSUMED collection. However, since the features and the data partitions used in these papers are different, direct comparisons between their experimental results might not be meaningful. Considering this, it will be desirable if we can have a 'standard' feature set and data partitioning scheme for the OHSUMED collection. This is the motivation of this paper.

2.2 The ".gov" collection

In TREC 2003 and 2004, there is a special track for web information retrieval, named the web track. The goal of this track is to investigate the retrieval behavior when the collection to be searched is in a large hyperlinked structure such as the World Wide Web. The web tracks used the ".gov" collection, which is based on a January, 2002 crawl of the ".gov" domain. There are in total 1,053,110 html documents in this collection, together with 11,164,829 hyperlinks.

Topic distillation is one of the tasks in the web track, which aims to find a list of entry points for good websites principally devoted to the topic. The focus is to return entry pages of good websites rather than the pages containing relevant information themselves, since the entry pages provides a better overview of the coverage of a topic in the collection. TREC committee provides judgment for the topic distillation task. The human assessors make binary judgments to as whether a page is appropriate to a given query. That is, a page is judged relevant only if it is the entry page of some website which is principally devoted to the query topic. In this regard, this task is very similar to the Web search scenario. There are 50 queries and 75 queries in topic distillation tasks of TREC 2003 and 2004, respectively.

Many research papers [27], [28], [33], [34] have been published using the topic distillation task on the ".gov" collection as their experimental platform. Similar to the situation for the OHSUMED collection, since the features and the data partitions are different in these papers, the corresponding experimental results are not directly comparable. This motivates us to extract 'standard' features and provide 'standard' data partitioning schemes for the ".gov" collection.

3. FEATURE EXTRACTION AND DATA PARTITIONING

When extracting features for the aforementioned purpose, we followed the principles as below:

(1) We try to cover as many classical features in IR as possible.

(2) We try to reproduce all the features proposed in recent SIGIR papers that used the OHSUMED collection or the ".gov" collection for their experiments.

(3) When extracting features, we conform to the original documents or papers. If the authors mentioned parameter tuning with regard to the feature, we also conducted parameter tuning. If the authors only provided a default parameter and have not mentioned parameter tuning, we used their default parameter directly in our feature extraction process.

3.1 Features for the OHSUMED collection

We extracted features from each judged query-document pair in the OHSUMED collection. We index the fields of *.T* (title) and *.W* (abstract) for documents and the field *.W* for queries. For both documents and queries, the field *.I* is used as id.

We extracted both 'low-level' and 'high-level' features from the OHSUMED collection. Low-level features include term frequency (*tf*), inverse document frequency (*idf*), document length (*dl*) and their combinations [1]. High-level features include the outputs of BM25 [29] and LMIR [35] algorithms. In particular, for LMIR, different smoothing methods (DIR, JM, ABS) [35] were utilized. In total, we extracted 25 features (10 from title, 10 from abstract, and 5 from 'title + abstract'). These features are summarized in Table 1 and Table 2. Note that we refer to those features proposed in recent SIGIR papers as "SIGIR feature".

3.2 Features for the TREC collection

Since there are many documents (webpages) in the ".gov" collection, when extracting features, we first used BM25 to rank all the documents with respect to each query, and then extracted features from the top 1000 documents. Considering that some relevant documents (judged by the TREC committee) may not appear in the top 1000 results according to their BM25 scores, we also extracted features from all the relevant documents for each query. The features can be classified into four categories.

Table 1. Low-level Features and their descriptions

Feature	Formulations	Descriptions	References
L1	$\sum_{q_i \in q \cap d} c(q_i, d)$	Term frequency (*tf*)	[1]
L2	$\sum_{q_i \in q \cap d} \log(c(q_i, d) + 1)$	SIGIR feature	[23]
L3	$\sum_{q_i \in q \cap d} \frac{c(q_i,d)}{\vert d \vert}$	Normalized *tf*	[1]
L4	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\vert d \vert} + 1\right)$	SIGIR feature	[23]
L5	$\sum_{q_i \in q \cap d} \log\left(\frac{\vert C \vert}{df(q_i)}\right)$	Inverse doc frequency (*idf*)	[1]
L6	$\sum_{q_i \in q \cap d} \log\left(\log\left(\frac{\vert C \vert}{df(q_i)}\right)\right)$	SIGIR feature	[23]
L7	$\sum_{q_i \in q \cap d} \log\left(\frac{\vert C \vert}{c(q_i,C)} + 1\right)$	SIGIR feature	[23]
L8	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\vert d \vert} \log\left(\frac{\vert C \vert}{df(q_i)}\right) + 1\right)$	SIGIR feature	[23]
L9	$\sum_{q_i \in q \cap d} c(q_i, d) \log\left(\frac{\vert C \vert}{df(q_i)}\right)$	*tf*idf*	[1]
L10	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\vert d \vert} \frac{\vert C \vert}{c(q_i,C)} + 1\right)$	SIGIR feature	[23]

Table 2. High-level Features and their descriptions

Feature	Descriptions	References
H1	BM25 score	[29]
H2	log(BM25 score)	[29]
H3	LMIR with DIR smoothing	[35]
H4	LMIR with JM smoothing	[35]
H5	LMIR with ABS smoothing	[35]

1) Low-level Content Features

Low-level content features include term frequency (*tf*), inverse document frequency (*idf*), document length (*dl*), and their combinations (e.g. *tf*idf*) [1]. For each of these features, we have four different values corresponding to the four fields of a webpage: body, anchor, title, and URL.

2) High-level Content Features

High-level content features include the outputs of BM25 [29] and LMIR [35] algorithms. We extracted four BM25 features in total, using the whole document, anchor text, title, and extracted title [17], respectively. For LMIR, different smoothing methods (DIR, JM, and ABS) and three fields (anchor, title, and extracted title) were used. As a result, there are nine language model features in total.

3) Hyperlink Features

Hyperlink features include PageRank [25], HITS [21], and their variations (HostRank [34], topical PageRank and topic HITS [24]). Since HITS and topical HITS have both authority and hub scores, there are 7 hyperlink features in total.

4) Hybrid Features

Hybrid features refer to those features containing both content and hyperlink information, including “hyperlink-based relevance propagation” [30] and “sitemap-based relevance propagation” [28].

In total, we extracted 44 features for each query-document pair. The complete feature list is shown in Table 3. Note that we refer to those features proposed in recent SIGIR papers as “SIGIR feature”.

Table 3. All the features for the TREC datasets

Feature	Descriptions	References
1	BM25	[27]
2	document length (dl) of body	[1]
3	dl of anchor	
4	dl of title	
5	dl of URL	
6	HITS authority	[21]
7	HITS hub	
8	HostRank (SIGIR feature)	[34]
9	Inverse document frequency (idf) of body	[1]
10	idf of anchor	
11	idf of title	
12	idf of URL	
13	Sitemap based feature propagation (SIGIR feature)	[28]
14	PageRank	[25]
15	LMIR.ABS of anchor	[35],
16	BM25 of anchor	
17	LMIR.DIR of anchor	[35]
18	LMIR.JM of anchor	
19	LMIR.ABS of extracted title (SIGIR feature)	
20	BM25 of extracted title (SIGIR feature)	[29]
21	LMIR.DIR of extracted title (SIGIR feature)	[35]
22	LMIR.JM of extracted title (SIGIR feature)	
23	LMIR.ABS of title	
24	BM25 of title	[29]
25	LMIR.DIR of title	[35]
26	LMIR.JM of title	
27	Sitemap based feature propagation (SIGIR feature)	[28]
28	tf of body	[1]
29	tf of anchor	
30	tf of title	
31	tf of URL	
32	tf*idf of body	
33	tf*idf of anchor	
34	tf*idf of title	
35	tf*idf of URL	
36	Topical PageRank (SIGIR feature)	[24]
37	Topical HITS authority (SIGIR feature)	
38	Topical HITS hub (SIGIR feature)	
39	Hyperlink base score propagation: weighted in-link (SIGIR feature)	[28] [30]
40	Hyperlink base score propagation: weighted out-link (SIGIR feature)	
41	Hyperlink base score propagation: uniform out-link (SIGIR feature)	
42	Hyperlink base feature propagation: weighted in-link (SIGIR feature)	
43	Hyperlink base feature propagation: weighted out-link (SIGIR feature)	
44	Hyperlink base feature propagation: uniform out-link (SIGIR feature)	

3.3 Feature Normalization

Since the absolute values of a feature for different queries might not be comparable, we conducted query-based normalization for each feature. Suppose there are $N^{(i)}$ documents $\{d_j^{(i)} \vert j = 1, \ldots, N^{(i)}\}$ with respect to query i in the dataset. A feature of document $d_j^{(i)}$ is represented as $x_j^{(i)}(j = 1, \dots, N^{(i)})$. Then after normalization, the feature will become

$$\frac{x_j^{(i)}-\min\{x_k^{(i)},k=1,\dots,N^{(i)}\}}{\max\{x_k^{(i)},k=1,\dots,N^{(i)}\}-\min\{x_k^{(i)},k=1,\dots,N^{(i)}\}}.$$

3.4 Dataset Partitioning

As aforementioned, in the LETOR dataset, we have three query sets: one for OHSUMED with 106 queries, one for the topic distillation task of TREC 2003 (TD2003) with 50 queries, and another for the topic distillation task of TREC 2004 (TD2004) with 75 queries. As a result, we actually have three subsets in the LETOR dataset for experimental studies: OHSUMED, TD2003, and TD2004.

We partitioned each subset into five parts, denoted as S1, S2, S3, S4, and S5, in order to conduct five-fold cross validation. For each fold, we used three parts for training, one part for validation, and the remaining part for testing (See Table 4). The training set is used to learn the ranking model. The validation set is used to tune the parameters of the ranking model, such as the number of iterations in RankBoost, and the combination coefficient in the objective function of Ranking SVM. The test set is used to report the ranking performance of the model. Note that since we conduct five-fold cross validation, the reported performance in this paper is actually the average over different folds.

The LETOR dataset, containing the aforementioned feature representations of documents, their relevance judgments with respective to queries, and the partitioned training, validation and test sets have been release at the website of Microsoft Research. It can be downloaded from http://research.microsoft.com/users/tyliu/LETOR/.

Table 4. Data Partitioning for 5-fold Cross Validation

Folds	Training set	Validation set	Test set
Fold1	{S1, S2, S3}	S4	S5
Fold2	{S2, S3, S4}	S5	S1
Fold3	{S3, S4, S5}	S1	S2
Fold4	{S4, S5, S1}	S2	S3
Fold5	{S5, S1, S2}	S3	S4

4. BASELINE EXPERIMENTAL RESULTS

4.1 Ranking Algorithms

4.1.1 Ranking SVM

Many previous studies have shown that Ranking SVM [16] [20] is an effective algorithm for ranking. Ranking SVM generalizes SVM to solve the problem of ranking: while traditional SVM works on documents, Ranking SVM adopts partial-order preference for document pairs as its constraints. The optimization formulation of Ranking SVM is as follows:

$$\min \frac{1}{2}w^T w + C\sum_{i,j,q}\varepsilon_{i,j,q},$$

$$s.t. \forall(d_i, d_j) \in r_q^*: \omega\phi(q, d_i) \geq \omega\phi(q, d_j) + 1 - \varepsilon_{i,j,q}.$$

When running Ranking SVM on the LETOR dataset, we use its linear version.

4.1.2 RankBoost

Freund, Y. *et al* [11] adopt the Boosting approach and propose the RankBoost algorithm to learn the ranking function. Similar to Ranking SVM, RankBoost operates on document pairs. Like all boosting algorithms, RankBoost trains one weak learner at each round of iteration, and combines these weak learners as the final ranking function. After each round, the document pairs are re-weighted: it decreases the weight of correctly ranked pairs and increases the weight of wrongly ranked pairs. The detailed algorithm is shown in Figure 1.

When running RankBoost on the LETOR dataset we define each weak learner on the basis of a feature. With a proper threshold, the weak learner has binary output, i.e., it takes values from {0, 1}.

```
Algorithm: RankBoost
Given: initial weight $D$ over $X \times X$
Initialize: $D_1=D$
For $t=1,\dots,T$:
    (a) Train weak learner using weight $D_t$.
    (b) Get week ranking $h_t$: X → R.
    (c) Choose $\alpha_t \in R$.
    (d) Update weight by
        $D_{t+1}(x_0,x_1)=\frac{D_t(x_0,x_1)\exp(\alpha_t(h_t(x_0)-h_t(x_1)))}{Z_t}$
where $Z_t=\sum_{x_0,x_1}D_t(x_0,x_1)\exp(\alpha_t(h_t(x_0)-h_t(x_1)))$.
Output the final ranking function: $H(x)=\sum_{t=1}^{T}\alpha_t h_t(x)$
```

Figure 1. The RankBoost algorithm

4.2 Evaluation Measures

For evaluation, we use precision at position *n*, mean average precision, and normalized discount cumulative gain as the measures. These measures are widely used in information retrieval. Their definitions are as follows.

4.2.1 Precision at position n (P@n)

Precision at *n* measures the relevance of the top *n* documents in the ranking result with respect to a given query:

$$P@n = \frac{\#\,relevant\ docs\ in\ top\ n\ results}{n}.$$

For example, if the top 10 documents returned for a query are {*relevant, irrelevant, irrelevant, relevant, relevant, relevant, irrelevant, irrelevant, relevant, relevant*}, then P@1 to P@10 values will be {1, 1/2, 1/3, 2/4, 3/5, 4/6, 4/7, 4/8, 5/9, 6/10} respectively. For a set of queries, we average the P@*n* values of all the queries to get the mean P@*n* value. Since P@*n* asks for binary judgments while there are three levels of relevance judgments in the OHSUMED collection, we simply regard "definitely relevant" as relevant and the other two levels as irrelevant when computing P@*n*.

4.2.2 Mean average precision (MAP)

For a single query, average precision is defined as the average of the P@*n* values for all relevant documents:

$$AP = \frac{\sum_{n=1}^{N}(P@n * rel(n))}{\#total\ relevant\ docs\ for\ this\ query},$$

where N is the number of retrieved documents, and $rel(n)$ is a binary function on the relevance of the n-th document:

$$rel(n) = \begin{cases} 1, if\ the\ n^{th}\ doc\ is\ relevant \\ 0, otherwise \end{cases}.$$

Similar to mean P@n, over a set of queries, we get MAP by averaging the AP values of all the queries.

4.2.3 Normalized discount cumulative gain (NDCG)

Note that P@n and MAP can only handle cases with binary judgment: "relevant" or "irrelevant". Recently, a new evaluation measure called Normalized Discount Cumulative Gain (NDCG) [18] [19] has been proposed, which can handle multiple levels of relevance judgments. While evaluating a ranking list, NDCG follows two rules:

1) Highly relevant documents are more valuable than marginally relevant document;
2) The lower ranking position a document (of any relevance level) has, the less valuable it is for the user, because it is less likely to be examined by the user.

According to the above rules, the NDCG value of a ranking list at position n is calculated as follow:

$$N(n) \equiv Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{\log(1+j)},$$

where $r(j)$ is the rating of the j-th document in the ranking list, and the normalization constant Z_n is chosen so that the perfect list gets a NDCG score of 1. In the above equation, we can see that $2^{r(j)} - 1$ is the gain (G) of the j-th document, $\frac{2^{r(j)}-1}{\log(1+j)}$ is the discounted gain (DG), $\sum_{j=1}^{n} \frac{2^{r(j)}-1}{\log(1+j)}$ is the discounted cumulative gain (DCG) at position n of the list, and finally $Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{\log(1+j)}$ is the normalized discounted cumulative gain (NDCG) at position n of the list, which is called NDCG@n.

Note that in order to calculate NDCG scores, we need to define the ratings of each document. For the TD2003 and TD2004 subsets, we define two ratings {0, 1} corresponding to "*irrelevant*" and "*relevant*"; and for the OHSUMED subset, we define three ratings {0, 1, 2}, corresponding to "*not relevant*", "*partially relevant*", and "*definitely relevant*" respectively.

4.3 Experimental Results

4.3.1 The OHSUMED subset

We list the ranking performances of Ranking SVM and RankBoost on the OHSUMED subset in Table 5.

From Table 5, we can see that Ranking SVM and RankBoost perform similarly on the OHSUMED subset. In more details, Ranking SVM performs as well as RankBoost in term of mean average precision, while for NDCG, RankBoost is better than Ranking SVM for NDCG@1 to NDCG@4, and worse for the remaining NDCG values.

Note that traditional IR models such as BM25 are within our features set. If we compare the performance of Ranking SVM and RankBoost with that of BM25 (which P@1 is 0.519, NDCG@1 is 0.399, and MAP is 0.425 is for the OHSUMED subset), we can find that by combining BM25 with other features, the learning to rank algorithms can significantly outperform the single feature of BM25. This validates the value of adopting the learning to rank approach to information retrieval.

Table 5. Ranking Performance of Ranking SVM and RankBoost for OHSUMED

(a) Precision at position *n*

Algorithms	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10
RankBoost	0.605	0.595	0.586	0.562	0.545	0.525	0.516	0.505	0.494	0.495
Ranking SVM	0.634	0.619	0.592	0.579	0.577	0.558	0.536	0.525	0.517	0.507

(b) Mean average precision

Algorithms	MAP
RankBoost	0.440
Ranking SVM	0.447

(c) NDCG at position *n*

Algorithms	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	NDCG@6	NDCG@7	NDCG@8	NDCG@9	NDCG@10
RankBoost	0.498	0.483	0.473	0.461	0.450	0.442	0.439	0.436	0.433	0.436
Ranking SVM	0.495	0.476	0.465	0.459	0.458	0.455	0.447	0.445	0.443	0.441

Table 6. Ranking Performance of Ranking SVM and RankBoost for TD2003

(a) Precision at position *n*

Algorithms	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10
RankBoost	0.260	0.270	0.240	0.230	0.220	0.210	0.211	0.193	0.182	0.178
Ranking SVM	0.420	0.350	0.340	0.300	0.264	0.243	0.234	0.233	0.218	0.206

(b) Mean average precision

Algorithms	MAP
RankBoost	0.212
Ranking SVM	0.256

(c) NDCG at position *n*

Algorithms	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	NDCG@6	NDCG@7	NDCG@8	NDCG@9	NDCG@10
RankBoost	0.260	0.280	0.270	0.272	0.279	0.280	0.287	0.282	0.282	0.285
Ranking SVM	0.420	0.370	0.379	0.363	0.347	0.341	0.340	0.345	0.342	0.341

Table 7. Ranking Performance of Ranking SVM and RankBoost for TD2004

(a) Precision at position *n*

Algorithms	P@1	P@2	P@3	P@4	P@5
RankBoost	0.480	0.447	0.404	0.347	0.323
Ranking SVM	0.440	0.407	0.351	0.327	0.291

Algorithms	P@6	P@7	P@8	P@9	P@10
RankBoost	0.304	0.293	0.277	0.262	0.253
Ranking SVM	0.273	0.261	0.247	0.236	0.225

(b) Mean average precision

Algorithms	MAP
RankBoost	0.383514
Ranking SVM	0.350459

(c) NDCG at position *n*

Algorithms	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5
RankBoost	0.480	0.473	0.464	0.439	0.437
Ranking SVM	0.440	0.433	0.409	0.406	0.393

Algorithms	NDCG@6	NDCG@7	NDCG@8	NDCG@9	NDCG@10
RankBoost	0.448	0.457	0.461	0.464	0.472
Ranking SVM	0.397	0.406	0.410	0.414	0.420

4.3.2 The TD 2003 subset

We list the ranking performance of Ranking SVM and RankBoost on the TD2003 subset in Table 6. From Table 6, we can see that Ranking SVM is better than RankBoost with all measures for TD2003. Take NDCG for example, Ranking SVM outperforms RankBoost by 0.15 (or 60% relatively) for NDCG@1, and more than 0.05 (or relatively 20%) for other NDCG values.

Also we can get the similar conclusion to that for the OHSUMED subset on the comparison between traditional ranking models and learning to rank methods. This time, for the TD2003 subset, the performance of BM25 is shown as follows: 0.12 for NDCG@1, 0.12 for P@1, and 0.126 for MAP. It is clear by combining more features, Ranking SVM and RankBoost can significantly outperform BM25.

4.3.3 The TD 2004 subset

The ranking performance of Ranking SVM and RankBoost on the TD2003 subset is shown in Table 7. From Table 7, we can see that RankBoost is slightly better than Ranking SVM for TD2004 in terms of all measure. And these two methods both outperform BM25 significantly. For the TD2004 subset, the performance of BM25 is as follows: 0.307 for NDCG@1, 0.307 for P@1, and 0.282 for MAP.

Overall speaking, the performance of Ranking SVM seems to be more stable than RankBoost and can outperform RankBoost in more cases. This on one hand shows that the large-margin nature of Ranking SVM really performs well on the ranking task. And on the other hand, this shows that the ranking task in the LETOR dataset is not so complex that linear models such as Ranking SVM have already done a good job, if the combination coefficients are carefully set. And the non-linear model of RankBoost does not necessarily lead to better performances.

5. FUTURE RESEARCH DIRECTIONS

So far, we have explained Ranking SVM and RankBoost, and shown the experimental results of them on the LETOR dataset. There are still many open problems with regard to learning to rank and one can study the issues using LETOR. We list several possible research topics here.

1) Feature selection for ranking

   It is obvious that the features we extracted are not equally effective. Thus, the LETOR dataset can be used to study different methods for features selection. To our knowledge, the work on feature selection for ranking is still an unsolved problem [1] and needs more investigations.

2) Loss functions for ranking

   The state-of-the-arts algorithms for learning to rank extend existing classification techniques to solve the ranking problem. However, considering the difference between ranking and classification (e.g. the difference in evaluation measures), one would conjecture that employing a new approach completely suitable for ranking is necessary. For example, it would be sensible to define novel loss functions that can better meet the evaluation requirements in IR, particularly, to introduce loss functions at query level.

3) Ranking models

   In this paper, we only tested the cases of using linear model and additive model as ranking models. It would be interesting to look how other ranking models work. For example, we may ask whether it is possible to define a ranking model that can exploit the inter-relationship between documents in IR ranking.

4) Query-based ranking

   Ranking SVM, RankBoost, and all the other existing models are query independent models. However, from the IR point of view, it seems better to employ query-dependent models. It would be interesting to see whether it is possible to partition the query space, and train different ranking models for different subspaces (different types of queries).

Certainly, besides the aforementioned four directions, there are still many problems one can explore with the LETOR data.

6. CONCLUSIONS

Research on machine learning is heavily affected by the availability of data, and this is also true for learning to rank for information retrieval. We believe that the LETOR dataset described in this paper has the potential to support future research on the area. On May 8, 2007, the LETOR dataset was released from Microsoft Research Asia. Up to the time this paper is written, there has been more than 250 downloads of this dataset, indicating that the need for such kind of data is really high.

We hope that by explaining the data creation process and providing the results of the state-of-the-arts algorithms in this paper, we can help people to better understand the nature of the dataset and to more effectively utilize the dataset in their research work. Finally, we hope that more datasets can be created and shared out from different groups after LETOR, and learning to rank for information retrieval can be significantly advanced.

REFERENCES

[1] Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval, Addison Wesley, 1999.

[2] Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G. Learning to Rank using Gradient Descent. Proceedings of the 22nd ICML, Bonn, Germany, 2005.

[3] Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W. Adapting ranking SVM to document retrieval. Proceedings of SIGIR2006, Seattle, USA.

[4] Chirita, P.A., Diederich, J., and Nejdl, W. MailRank: using ranking for spam detection, Proceedings of the 14th ACM international conference on Information and knowledge management, 2005.

[5] Chu, W., and S. Sathiya Keerthi, New approaches to support vector ordinal regression, Proceedings of the 22nd ICML, Bonn, Germany, 2005.

[6] Crammer, K., & Singer, Y. (2002). Pranking with ranking. NIPS 14.

[7] Cohen, W.W., Schapire, R.E., and Singer, Y. Learning to order things, Proceedings of the 1997 conference on Advances in neural information processing systems 10, p.451-457, July 1998, Denver, Colorado, United States

[8] Collins, M. Ranking algorithms for named-entity extraction: boosting and the voted perceptron, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, July 07-12, 2002, Philadelphia, Pennsylvania.

[9] Dave, K., Lawrence, S., Pennock, D.M. Mining the peanut gallery: opinion extraction and semantic classification of product reviews, Proceedings of the 12th international conference on World Wide Web, May 20-24, 2003, Budapest, Hungary.

[10] David D. Lewis, Yiming Yang, Tony G. Rose, Fan Li, RCV1: A New Benchmark Collection for Text Categorization Research, The Journal of Machine Learning Research, vol.5, Pages: 361 - 397, 2004.

[11] Freund, Y., Iyer, R., Schapire, R., & Singer, Y., An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, 2003 (4).

[12] Geng, X. B., Liu, T. Y., Qin, T., and Li, H. Feature Selection for Ranking. Proceedings of the 30th Annual International ACM SIGIR Conference, 2007. to appear.

[13] Gyongyi, Z., Garcia-Molina, H., and Pedersen, J. Combating web spam with TrustRank. In Proceedings of the 30th VLDB Conference, Sept. 2004.

[14] Harrington, E. F. Online Ranking/Collaborative filtering using the Perceptron Algorithm. Proceedings of the 20th International Conference on Machine Learning, pages 250-257, 2003.

[15] Hersh, W. R., Buckley, C., Leone, T. J., and Hickam, D. H. OHSUMED: An interactive retrieval evaluation and new large test collection for research. Proceedings of the 17th Annual ACM SIGIR Conference, pages 192-201, 1994.

[16] Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. Advances in Large Margin Classiers, MIT Press, Pages: 115-132.

[17] Hu, Y. H., Xin, G. M., Song, R. H., Hu, G. P., Shi, S. M., Cao, Y. B., Li, H. Title extraction from bodies of HTML documents and its application to web page retrieval, Proceedings of SIGIR 2005.

[18] Jarvelin, K., & Kekalainen, J. (2000). IR evaluation methods for retrieving highly relevant documents. Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, p.41-48, July 24-28, 2000, Athens, Greece

[19] Jarvelin, K., & Kekalainen, J. Cumulated Gain-Based Evaluation of IR Techniques, ACM Transactions on Information Systems, 2002.

[20] Joachims, T. Optimizing Search Engines Using Clickthrough Data, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.

[21] Kleinberg, J. Authoritative sources in a hyperlinked environment. Journal of the ACM, Vol. 46, No. 5, 604-622, 1999.

[22] Matveeva, I., Burges, C., Burkard, T., Laucius, A., and Wong, L. High Accuracy Retrieval with Multiple Nested Ranker. Proceedings of SIGIR2006, Seattle, USA.

[23] Nallapati, R. Discriminative models for information retrieval. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, July 25-29, 2004, Sheffield, United Kingdom.

[24] Nie, L., Davison, B. D., Qi, X., Topical link analysis for web search, Proceedings of SIGIR 2006.

[25] Page, L., Brin, S., Motwani, R., and Winograd, T. The PageRank citation ranking: bringing order to the Web, Technical report, Stanford University, 1998.

[26] Pang, B., and Lee, L. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. ACL 2005

[27] Qin, T., Liu, T. Y., Lai, W., Zhang, X. D., Wang, D. S., and Li, H. Ranking with Multiple Hyperplanes. Proceedings of the 30th Annual International ACM SIGIR Conference, 2007. to appear.

[28] Qin, T., Liu, T. Y., Zhang, X. D., Chen, Z., and Ma, W. Y. A study of relevance propagation for web search. Proceedings of SIGIR 2005.

[29] Robertson, S. E. Overview of the okapi projects, Journal of Documentation, Vol. 53, No. 1, 1997, pp. 3-7.

[30] Shakery, A., Zhai, C. X. Relevance Propagation for Topic Distillation UIUC TREC 2003 Web Track Experiments, Proceedings of TREC 2003.

[31] Tsai, M. F., Liu, T. Y., Qin, T., Chen, H. H., Ma, W. Y. FRank: A Ranking Method with Fidelity Loss. Proceedings of the 30th Annual International ACM SIGIR Conference, 2007. to appear.

[32] Xu, J., Cao, Y.B., Li, H., Zhao, M. Ranking definitions with supervised learning methods, Special interest tracks and posters of the 14th international conference on World Wide Web, May 10-14, 2005, Chiba, Japan.

[33] Xu, J. and Li, H. AdaRank: A Boosting Algorithm for Information Retrieval. Proceedings of the 30th Annual International ACM SIGIR Conference, 2007. to appear.

[34] Xue, G. R., Yang, Q., Zeng, H. J., Yu, Y., and Chen, Z. Exploiting the hierarchical structure for link analysis. Proceedings of SIGIR 2005.

[35] Zhai, C. and Lafferty, J. A study of smoothing methods for language models applied to Ad Hoc information retrieval. Proceedings of SIGIR 2001, pp. 334-342, 2001.

[36] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[37] ftp://ftp.sanbi.ac.za/STACK/benchmarks/