LETOR: A Benchmark Collection for Learning to Rank for Information Retrieval

by Microsoft Research Asia, an incomplete draft

Editor: blank

Abstract

Learning to rank has attracted great attention recently in both information retrieval and machine learning communities. However, the lack of public dataset had stood in its way until the LETOR benchmark dataset (actually a group of three datasets) was released in the SIGIR 2007 workshop on Learning to Rank for Information Retrieval (LR4IR 2007). Since then, this dataset has been widely used in many learning to rank papers, and has greatly speeded up the corresponding research. Recently, we released the latest version, LETOR3.0. LETOR3.0 makes a lot of improvement over previous two versions, and it is more useful and reliable. In this paper, we describe the details of datasets in LETOR3.0, including collection information, query sets and implementation of feature extraction. We also benchmark several widely used learning to rank methods on LETOR3.0, illustrating the results of these methods, suggesting new directions for research, and providing baseline results for future study.

Keywords: learning to rank, information retrieval, document sampling, feature extraction,

1. Introduction

Ranking is the central problem for many information retrieval (IR) applications. These include document retrieval (Cao et al., 2006), collaborative filtering (Harrington, 2003), key term extraction (Collins, 2002), definition finding (Xu et al., 2005), important email routing (Chirita et al., 2005), sentiment analysis (Pang and Lee, 2005), product rating (Dave et al., 2003), and anti web spam (Gyöngyi et al., 2004). 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 and applied to IR applications. 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 [1] and RCV1 (Lewis et al., 2004) for text categorization, and SANBI EST [2] for clustering. Therefore, it would be very beneficial, if benchmark datasets for learning to rank could be developed.

To tackle this problem, we released a benchmark collection, LETOR (LEarning TO Rank, version 1.0), at the beginning of 2007 for learning to rank research. Since then, this dataset has been widely used in many learning to rank papers (Qin et al., 2008b; Xia et al., 2008; Elsas et al., 2008; Jin et al., 2008; Yeh et al., 2007; Xu et al., 2008a; Guiver and Snelson, 2008; Geng et al., 2008). LETOR1.0 was used in SIGIR2007 workshop on learning to rank for information retrieval [3]. At the end of 2007, we released a new version of LETOR, LETOR2.0, which was used in SIGIR2008 workshop on learning to rank for information retrieval [4]. In these two years, we collected many valuable feedbacks and suggestions about how to further improve LETOR. We released LETOR3.0 considering these feedbacks recently. LETOR3.0 increase many new information for learning to rank research, including 450 new queries associated with documents and features, meta features which can be used to derive some strong features, and more baseline learning to rank algorithms.

This paper attempts to provide the necessary documentation about the creation of LETOR (especially LETOR3.0) to ease the usage of this benchmark. We begin Section 2 by introducing learning to rank problem in information retrieval domain. Section 3 describe two original document collection (OHSUMED and ".Gov") and corresponding query sets used in LETOR. We discuss documents sampling strategy for a query in Section 4. Section 5 describes features we extracted for learning to rank algorithms, and Section 6 describes meta features which can be used to derive strong features for learning. Section 7 describes our data partition strategy in experiments and several learning to rank algorithms. The results of these algorithms are reported in Section 8. We discusses potential research directions for learning to rank in Section 9. The last section concludes the paper.

2. Learning to Rank for Information Retrieval

Learning to rank, when applied to information retrieval, is a task as follows. Assume that there is a collection of documents. In retrieval (i.e., ranking), given a query, the ranking function assigns a score to each document, and ranks the documents in descending order of the scores. The ranking order represents the relevance of documents with respect to the query. In learning, a number of queries are provided; each query is associated with a perfect ranking list of documents; a ranking function is then created using the training data, such that the model can precisely predict the ranking lists in the training data.

Due to its importance, learning to rank has been drawing broad attention in both machine learning community and information retrieval community, and many learning algorithms have been proposed recently. Roughly speaking, there are mainly two kinds of learning based approaches for ranking: pairwise approach and listwise approach. For other

1. http://www.daviddlewis.com/resources/testcollections/reuters 21578/
2. ftp://ftp.sanbi.ac.za/STACK/benchmarks/
3. http://research.microsoft.com/users/LR4IR-2007/
4. http://research.microsoft.com/users/LR4IR-2008/

approaches, see Shashua and Levin (2002); Crammer and Singer (2001); Lebanon and Lafferty (2002), for example.

Pairwise approach takes document pairs as instances in learning, and formalizes the problem of learning to rank as that of classification. Specifically, in learning it collects document pairs from the ranking lists, and for each document pair it assigns a label representing the relative relevance of the two documents. It then trains a classification model with the labeled data and makes use of the classification model in ranking. The uses of Support Vector Machines (SVM), Boosting, and Neural Network as the classification model lead to the methods of Ranking SVM (Herbrich et al., 1999), RankBoost (Freund et al., 2003), and RankNet (Burges et al., 2005). Many algorithms have been proposed to further improve pairwised algorithms, such as FRank (Tsai et al., 2007), multiple hyperplane ranker (Qin et al., 2007a) and nested ranker (Matveeva et al., 2006).

Listwise approach takes document lists as instance in learning and the loss function is defined on that basis. Representative work include ListNet (Cao et al., 2007), RankCosine (Qin et al., 2007b), ListMLE (Xia et al., 2008). A sub branch of listwise approach is direct optimization of IR measures, such as AdaRank (Xu and Li, 2007), SoftRank (Taylor et al., 2008), SVMMap (Yue et al., 2007) and PermuRank (Xu et al., 2008b).

To conduct experiments for learning to rank, one first need to survey to hunt a document collection, then index the document collection, extract feature vector representation for each query-document pair, and finally the performance of a algorithm can be verified. Considering that purchase of a document collection may be costly, collection indexing needs good understanding of search platform and feature extraction needs detailed knowledge of information retrieval, it is very difficult for individual researcher to create a dataset. To reducing the entry cost of learning to rank research, we released LETOR, a benchmark collection. The value of the LETOR dataset lies in the following two aspects:

1. LETOR contains a set of standard features, and so researchers can directly use them to test the effectiveness of their ranking algorithms, without the necessity of creating a dataset by their own. Because the dataset creation is very costly (including collection hunting, query selection, feature extraction, etc.), LETOR has greatly reduced the barrier of the research on learning to rank.

2. LETOR makes the fair comparison among different learning to rank algorithms possible. Before the release of LETOR, different datasets (i.e. different query sets, different document collections, different features, or different evaluation tools) were used in different papers. As a result, it is not easy to get conclusive observations on the effectiveness of the learning to rank algorithms, by comparing the performances reported in different papers. LETOR has made it possible to compare the algorithms, since a standard dataset and a set of standard evaluation tools are used. Furthermore, in the current version of LETOR (version 3.0), several state-of-the-art baselines have been included, which even further ease the algorithm comparison: researchers even do not need to implement some of the baselines to be compared in their work.

3. Document Collections

Given a query, a retrieval system should return a number of relevance documents from some document collection. In LETOR, we used two document collections: .Gov and OHSUMED.

3.1 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.

There are three search tasks in TREC Web track: topic distillation, homepage finding and named page finding

Topic distillation 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.

Homepage finding task is required to return the homepage of the query, and named page finding task aims to return the page whose name is exactly the query. Generally speaking, there is only one answer for homepage finding query and named page finding query.

As an example of differentiating these three tasks, we cite the TREC guideline as follows. Consider the query USGS, which is the acronym for the US Geological Survey. Some interpretations of this query are:

- Topic distillation: Find me homepages of sites describing aspects of the US Geological Survey. Possible answers include http://www.usgs.gov, http://water.usgs.gov, http://geography.usgs.gov, http://earthquake.usgs.gov. (Another example: literacy, might mean find me .GOV sites about literacy, with answers including http://www.nifl.gov/, http://nces.ed.gov/naal/ and http://nces.ed.gov/surveys/all/.)

- Homepage finding: Find me the URL of the USGS homepage (http://www.usgs.gov). I have forgotten or do not know that URL, or I prefer typing usgs to typing the full URL.

- Named page finding: Find me the URL of a non-homepage, e.g., searching for http://www.usgs.gov/aboutusgs.html with the query introduction to usgs. The query is the name of the page in question (rather than, e.g., words describing its topic).

The statistics of queries for three tasks are shown as Table 1. In LETOR2.0, we only extracted features for queries of topic distillation task. We extracted features for all queries of the three tasks. And so LETOR3.0 contains 450 more queries than LETOR2.0.

Table 1: Number of queries in TREC web track

Task	TREC2003	TREC2004
Topic distillation	50	75
Homepage finding	150	75
Named page finding	150	75

Many research papers (**?**Xu and Li, 2007; Qin et al., 2005; Xue et al., 2005) have been published using the topic distillation task on the .Gov collection as their experimental platform. 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.

3.2 The OHSUMED collection

The OHSUMED collection (Hersh et al., 1994) 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 (**?**Xu and Li, 2007) have been published using the OHSUMED collection. Similar to the situation for the .Gov collection, 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 LETOR.

4. Document Sampling

For a learning to rank algorithm, we need some query set and each query associated with some documents labeled as relevant/irrelevant.[5] Due to the large scale of document collection, it is unpractical to check every document and judge whether it is relevant to a given query. In most information retrieval settings, given a query, only some "possible" documents are selected for labeling. Similarly, for a query, it is not practical to extract feature vectors of all the documents. A reasonable strategy is to sample some "possible" relevant documents, and then extract feature vectors for these query-document pairs. In this section, we discuss the sampling strategy used in LETOR.

4.1 Sampling Strategy for .Gov

For .Gov collection, given a query, TREC committee labels some documents as relevant and some as irrelevant. For remaining unlabeled documents, TREC committee treat them as irrelevant in evaluation. And so in LETOR creation, we also treat the unlabeled documents for a query as irrelevant.

In LETOR2.0, we sampled documents for a query as follows. 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. That is, the document pool for a query include (a) the top 1000 documents ranked by BM25 plus (b) any other documents judged relevant.

Minka and Robertson (2008) argued that such a sampling strategy is biased: "documents with high BM25 scores are selected anyway, irrespective of relevance; but documents with low BM25 scores are selected only if they are relevant." They found that in LETOR2.0, BM25 is even negatively correlated with relevance. Considering this, we only selected the top 1000 documents by BM25 for each query in LETOR3.0. Note that for some rare query, it is possible that less than 100 documents contain at least one query term, and so there may be some queries which has less than 1000 associated documents.

4.2 Sampling Strategy for OHSUMED

For OHSUMED collection, we adopted a same sampling strategy for all three versions (1.0, 2.0, 3.0) of LETOR: given a query, only judged documents were selected for feature extraction. The judgments are "highly relevant", "partially relevant" or "irrelevant". According such a sampling strategy, a query has about 152 documents on average for feature extraction (Qin et al., 2008a).

We would like to point out that such a sampling strategy is not perfect. As pointed out by Minka and Robertson (2008) after the release of LETOR2.0, "this selection meant that documents in the pool had a high chance of being relevant. This is evident from the

5. There may be multiple levels of relevance judgement, such as that of OHSUMED corpus.

proportion of non-relevants among the selected documents nine topics have less than 40% in the non-relevant category. Furthermore, every document in the pool matched some version of the query very well. Thus these non-relevant documents are highly atypical. Examples of the wider range of non-relevant documents that clearly exist in the full collection are missing in the dataset. In particular, there are likely to be many other documents that could be scored relatively highly by a ranking algorithm (which one would particularly like the algorithm to learn to distinguish)."

A possible solution to overcome this problem is that for a query, we include all the documents containing at least one query term. By doing so, we can signicantly increase the number of documents to be ranked. However, there are also some concerns with this method because we have introduced a number of unjudged document into the dataset. First, in training, it is not clear how to use these un-judged documents. Simply regarding them as irrelevant may or may not be reasonable. In fact, this is a research problem by itself: how to make good use of unlabeled documents in learning. Second, in testing, it is difficult to evaluate the accuracy of a ranking algorithm. At least the current evaluation measures used in LETOR, MAP and NDCG, cannot be computed in this case. Considering the two issues, we keep the same sampling strategy in LETOR3.0 as that in LETOR2.0.

5. Features for Learning

In this section, we introduce the features in LETOR3.0 which can be directly used by learning algorithms. The feature file is formatted in a matrix style. Each row represents a query-document pair. The first column is relevance judgement of the query-document pair, the second column is query id, followed with feature id and feature values, and the last column is the document id.

5.1 Features of .Gov Collection

For .Gov collection, there are 44 features in LETOR2.0. In LETOR3.0 we added in-link number, out-link number, length of URL, number of slashes in the URL, etc. as new features. Also, we extracted those existing features in all streams (URL, title, anchor and body), while features in some streams are missing in LETOR2.0. Overall, there are be 64 features (Table. 2) which can be directly used by learning algorithms.

Table 2: Learning Features of TREC

ID	Feature Description
1	Term frequency (TF) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	Inverse document frequency (IDF) of body
7	IDF of anchor
8	IDF of title

9	IDF of URL
10	IDF of whole document
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
16	Document length (DL) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of whole document
36	LMIR.JM of body
37	LMIR.JM of anchor
38	LMIR.JM of title
39	LMIR.JM of URL
40	LMIR.JM of whole document
41	Sitemap based term propagation
42	Sitemap based score propagation
43	Hyperlink base score propagation: weighted in-link
44	Hyperlink base score propagation: weighted out-link
45	Hyperlink base score propagation: uniform out-link
46	Hyperlink base feature propagation: weighted in-link
47	Hyperlink base feature propagation: weighted out-link
48	Hyperlink base feature propagation: uniform out-link
49	HITS authority
50	HITS hub
51	PageRank

52	HostRank
53	Topical PageRank
54	Topical HITS authority
55	Topical HITS hub
56	Inlink number
57	Outlink number
58	Number of slash in URL
59	Length of URL
60	Number of child page
61	BM25 of extracted title
62	LMIR.ABS of extracted title
63	LMIR.DIR of extracted title
64	LMIR.JM of extracted title

Now we give some details of these features and also list out the differences between the implementation of some features in LETOR3.0 and that in LETOR2.0.

1. We adopted a new parser while indexing documents. This may cause term frequency related features are different from that in LETOR2.0. A direct result is that the BM25 score of a document in LETOR3.0 is different from that in LETOR2.0, and so the top 1000 document set of a query by BM25 in LETOR3.0 may be different from that in LETOR2.0.

2. In LETOR3.0, we considered 5 streams of a document: body, anchor, title, URL and whole document. The last one is the union of the former four ones. Here we would like to point out that the body stream in LETOR3.0 contains more content than in LETOR2.0, such as the text in "meta" tag and "alt" tag of a web page. And so in LETOR2.0, the union of body, anchor, title and URL is not equal to the whole document, since the text in some tags are missing.

3. For term frequency feature (e.g. feature 1-5), its value is the sum of term frequency of each query term. Comparing with LETOR2.0, we added a new feature, term frequency of whole document, in LETOR3.0, and its value is equal to the sum of term frequency of body, anchor, title and URL.

4. We computed the IDF (inverse document frequency) of query term q_i as follows in LETOR2.0:
$$idf(q_i) = \frac{1}{n(q_i)},$$
where $n(q_i)$ is the number of documents containing q_i. For a stream (e.g. title), $n(q_i)$ is the number of documents containing q_i in the stream (e.g. title). In LETOR3.0, we adopted a widely used computation:
$$idf(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}, \quad (1)$$

where N is the total number of documents in the collection. Since a query may contains more than one term, the final value of IDF in LETOR3.0 is the sum of the IDF of each query term:

$$idf(Q) = \sum_{q_i} idf(q_i).$$

Note that IDF is document independent, and so all the documents under a query have same IDF values.

5. Given a query Q, containing keywords $q_1, ..., q_t$, the BM25 score of a document D is computed as:

$$BM25(D,Q) = \sum_{i:f(q_i,D)>0} idf(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot(1-b+b\cdot\frac{|D|}{avgdl})} \cdot \frac{(k_3+1)f(q_i,Q)}{k_3+f(q_i,Q)}, \quad (2)$$

where $idf(q_i)$ is defined as in Eq. (1), $f(q_i, D)$ is the occurrences of q_i in the document D, $f(q_i, Q)$ is the occurrences of q_i in the query Q, $|D|$ is the length of the document D (i.e., the number of words), and $avgdl$ is the average document length in the entire document collection from which documents are drawn. k_1, k_3 and b are free parameters. In feature 21-25 and 61, we set $k_1 = 2.5$, $k_3 = 0$ and $b = 0.8$. The $avgdl$ for each stream can be gotten from meta features, as described in next section.

6. For language model related features (26-40, 62-64), we followed the implementation of Zhai and Lafferty (2001) and used the parameters suggested in their paper: for LMIR.ABS features, we set its parameter $\delta = 0.7$; for LMIR.DIR features, we set its parameter $\mu = 2000$; for LMIR.JM features, we set its parameter $\lambda = 0.1$.

7. For sitemap based propagation features (41-42), we set the propagation rate $\alpha = 0.9$ (Qin et al., 2005). We find that the outputs of these two features are very similar. For hyperlink based propagation features (43-48), we set the propagation rate $\alpha = 0.85$ (Shakery and Zhai, 2003).

8. Because the original PageRank score of a document is very small, we multiplied it by 10^5. We also multiplied the original HostRank score by 10^3.

9. For topical PageRank and topical HITS, we used the same twelve categories as in Nie et al. (2006). [6]

10. To count the number of child page of a web page (feature 60), we simply used the URLs of documents to recover parent-child relationship between any two web pages.

5.2 Features of OHSUMED Collection

For OHSUMED collection, there are 25 features in LETOR2.0. We extracted 20 new features from the fields of title, abstract, and title + abstract: 10 low level features from title + abstract, 5 high level features from title, and 5 high level features from abstract. The entire feature set can be found in Table 3.

6. We'd like to thanks Lan Nie, Brian D. Davison and Xiaoguang Qi for providing us the models for web page classification.

Table 3: Learning Features of OHSUMED

ID	Feature Description
1	$\sum_{q_i \in q \cap d} c(q_i, d)$ in 'title'
2	$\sum_{q_i \in q \cap d} \log\left(c(q_i, d) + 1\right)$ in 'title'
3	$\sum_{q_i \in q \cap d} \frac{c(q_i,d)}{\lvert d \rvert}$ in 'title'
4	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\lvert d \rvert} + 1\right)$ in 'title'
5	$\sum_{q_i \in q \cap d} \log\left(\frac{\lvert C \rvert}{df(q_i)}\right)$ in 'title'
6	$\sum_{q_i \in q \cap d} \log\left(\log\left(\frac{\lvert C \rvert}{df(q_i)}\right)\right)$ in 'title'
7	$\sum_{q_i \in q \cap d} \log\left(\frac{\lvert C \rvert}{c(q_i,C)} + 1\right)$ in 'title'
8	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\lvert d \rvert} \cdot \log\left(\frac{\lvert C \rvert}{df(q_i)}\right) + 1\right)$ in 'title'
9	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot \log\left(\frac{\lvert C \rvert}{df(q_i)}\right)$ in 'title'
10	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\lvert d \rvert} \cdot \frac{\lvert C \rvert}{c(q_i,C)} + 1\right)$ in 'title'
11	BM25 score in 'title'
12	log(BM25 score) in 'title'
13	LMIR with DIR smoothing in 'title'
14	LMIR with JM smoothing in 'title'
15	LMIR with ABS smoothing in 'title'
16	$\sum_{q_i \in q \cap d} c(q_i, d)$ in 'abstract'
17	$\sum_{q_i \in q \cap d} \log\left(c(q_i, d) + 1\right)$ in 'abstract'
18	$\sum_{q_i \in q \cap d} \frac{c(q_i,d)}{\lvert d \rvert}$ in 'abstract'
19	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\lvert d \rvert} + 1\right)$ in 'abstract'
20	$\sum_{q_i \in q \cap d} \log\left(\frac{\lvert C \rvert}{df(q_i)}\right)$ in 'abstract'
21	$\sum_{q_i \in q \cap d} \log\left(\log\left(\frac{\lvert C \rvert}{df(q_i)}\right)\right)$ in 'abstract'
22	$\sum_{q_i \in q \cap d} \log\left(\frac{\lvert C \rvert}{c(q_i,C)} + 1\right)$ in 'abstract'
23	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\lvert d \rvert} \cdot \log\left(\frac{\lvert C \rvert}{df(q_i)}\right) + 1\right)$ in 'abstract'
24	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot \log\left(\frac{\lvert C \rvert}{df(q_i)}\right)$ in 'abstract'
25	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\lvert d \rvert} \cdot \frac{\lvert C \rvert}{c(q_i,C)} + 1\right)$ in 'abstract'
26	BM25 score in 'abstract'
27	log(BM25 score) 'abstract'
28	LMIR with DIR smoothing in 'abstract'
29	LMIR with JM smoothing in 'abstract'
30	LMIR with ABS smoothing in 'abstract'
31	$\sum_{q_i \in q \cap d} c(q_i, d)$ in 'title + abstract'
32	$\sum_{q_i \in q \cap d} \log\left(c(q_i, d) + 1\right)$ in 'title + abstract'
33	$\sum_{q_i \in q \cap d} \frac{c(q_i,d)}{\lvert d \rvert}$ in 'title + abstract'

34	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\|d\|} + 1\right)$ in 'title + abstract'
35	$\sum_{q_i \in q \cap d} \log\left(\frac{\|C\|}{df(q_i)}\right)$ in 'title + abstract'
36	$\sum_{q_i \in q \cap d} \log\left(\log\left(\frac{\|C\|}{df(q_i)}\right)\right)$ in 'title + abstract'
37	$\sum_{q_i \in q \cap d} \log\left(\frac{\|C\|}{c(q_i,C)} + 1\right)$ in 'title + abstract'
38	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\|d\|} \cdot \log\left(\frac{\|C\|}{df(q_i)}\right) + 1\right)$ in 'title + abstract'
39	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot \log\left(\frac{\|C\|}{df(q_i)}\right)$ in 'title + abstract'
40	$\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i,d)}{\|d\|} \cdot \frac{\|C\|}{c(q_i,C)} + 1\right)$ in 'title + abstract'
41	BM25 score in 'title + abstract'
42	log(BM25 score) 'title + abstract'
43	LMIR with DIR smoothing 'title + abstract'
44	LMIR with JM smoothing in 'title + abstract'
45	LMIR with ABS smoothing in 'title + abstract'

Now we give some details of these features.

1. For language model related features (13-15, 28-30, 43-45), we followed the implementation of Zhai and Lafferty (2001) and used the parameters suggested in their paper: for LMIR.ABS features, we set its parameter $\delta = 0.5$; for LMIR.DIR features, we set its parameter $\mu = 50$; for LMIR.JM features, we set its parameter $\lambda = 0.5$.

2. The computation of BM25 is same as Eq. (2). We set parameters as $k_1 = 1.2$, $k_3 = 7$ and $b = 0.75$.

6. Meta Features

In previous section, some features we extracted are good ranking models by themselves, such as BM25 and LMIR features, and these features have their own parameters. One may want to re-produce these features and tune their parameters. For this purpose, LETOR3.0 also provides many meta features, which can be used to derive some strong features. In this section, we describe the meta features contained in LETOR3.0.

There are two kinds of meta features provided in LETOR3.0: one is about statistic information of a collection, and the other one is about raw information of the documents associated with each query.

The information about the collection is contained in a separate xml file, as in Figure. 1. As can be seen, this meta file contains the document number, stream number and description, and the (unique) term number in each stream. One can get the *avgdl* of each stream from these information.

For each individual query, there is a separate xml file (as Figure. 2) about the raw information of itself and its associated documents. Line 3 indicates the id of the query, which comes from original collection. As can be seen, there are three big components: streaminfo, terminfo, and docinfo.

```
<?xml version="1.0" encoding="utf-8"?>
<collectioninfo>
  <name>OHSUMED</name>
  <docnum>348566</docnum>
  <streamnum>3</streamnum>
  <stream>
    <streamid>1</streamid>
    <streamdescription>Title</streamdescription>
    <uniquetermnum>53613</uniquetermnum>
    <totaltermnum>2770138</totaltermnum>
  </stream>
  <stream>
    <streamid>2</streamid>
    <streamdescription>Abstract</streamdescription>
    <uniquetermnum>120208</uniquetermnum>
    <totaltermnum>23680877</totaltermnum>
  </stream>
  <stream>
    <streamid>3</streamid>
    <streamdescription>Title and Abstract</streamdescription>
    <uniquetermnum>126335</uniquetermnum>
    <totaltermnum>26451015</totaltermnum>
  </stream>
</collectioninfo>
```

(a) OHSUMED collection

```
<?xml version="1.0" encoding="utf-8" ?>
<collectioninfo>
  <name>GOV</name>
  <docnum>1053111</docnum>
  <streamnum>5</streamnum>
  <stream>
    <streamid>1</streamid>
    <streamdescription>body</streamdescription>
    <uniquetermnum>2700970</uniquetermnum>
    <totaltermnum>482663137</totaltermnum>
  </stream>
  <stream>
    <streamid>2</streamid>
    <streamdescription>anchor</streamdescription>
    <uniquetermnum>152739</uniquetermnum>
    <totaltermnum>4942973</totaltermnum>
  </stream>
  <stream>
    <streamid>3</streamid>
    <streamdescription>title</streamdescription>
    <uniquetermnum>121029</uniquetermnum>
    <totaltermnum>4833634</totaltermnum>
  </stream>
  <stream>
    <streamid>4</streamid>
    <streamdescription>url</streamdescription>
    <uniquetermnum>409732</uniquetermnum>
    <totaltermnum>7476931</totaltermnum>
  </stream>
  <stream>
    <streamid>5</streamid>
    <streamdescription>whole document</streamdescription>
    <uniquetermnum>2926310</uniquetermnum>
    <totaltermnum>499916675</totaltermnum>
  </stream>
</collectioninfo>
```


(b) .Gov collection

Figure 1: Colletion information

```xml
<?xml version="1.0" encoding="utf-8"?>
<query>
  <id>xxx</id>

  <streaminfo>
    <stream>
      <streamid>1</streamid>
      <streamspecificquerylength>4</streamspecificquerylength>
    </stream>
  </streaminfo>

  <terminfo>
    <termnum>xxx</termnum>
    <term>
      <termid>xxx</termid>
      <word>xxx</word>
      <stream>
        <streamid>xxx</streamid>
        <streamfrequency>xxx</streamfrequency>
        <streamtermfrequency>xxx</streamtermfrequency>
      </stream>
    </term>
  </terminfo>

  <docinfo>
    <docnum>xxx</docnum>
    <doc>
      <docid>xxx</docid>
      <label>xxx</label>
      <stream>
        <streamid>xxx</streamid>
        <length>xxx</length>
        <uniquelength>xxx</uniquelength>
        <term>
          <termid>xxx</termid>
          <termfrequency>xxx</termfrequency>
        </term>
      </stream>
    </doc>
  </docinfo>

</query>
```

Figure 2: XML file to describe a query

- The node "streaminfo" (line 5-10 in Figure 2) describes statistical information of query terms with respect to each stream. Here the node "streamid" is consistent with id in the XML file of collection information. We say that "a term appears in a stream (e.g. title) of some collection" if this term is contained in the stream (e.g. title) of at least one document in this collection. "streamspecificquerylength" means how many query terms appears in this specific stream of the collection. For example, this query may contain 4 terms, and only 2 terms of them appears in title stream of this collection. This information is needed by language model based ranking models. If there are n streams in this collection, then there will be n "stream" nodes under "streaminfo" node.

- The node "terminfo" (line 12-23 in Figure 2) contains information of each query term with respect to every individual stream. [7] The node "termnum" indicates total term number contained in this query. The node "streamfrequency" indicates how many documents in the collection containing query term "work" in stream "streamid". The node "streamtermfrequency" indicates how many times query term "word" appears in stream "streamid" of all the documents in this collection. A node "term" may contain multiple "stream" as child nodes, which depends on the number of stream in the collection; a node "terminfo" may have multiple "term" as child nodes, depending on the number of terms in the query.

- The node "docinfo" (line 25-40 in Figure 2) plays a role similar to forward index, but limits to query terms and the selected documents. The node "docnum" indicates how many documents selected for this query, followed by information of these selected documents. Here "docid" is consistent with that in learning based feature files. The node "label" means the judgement of the document w.r.t the query. The node "length" under the node "stream" means the total words (not limited by query terms) in this stream of this document, and the node "uniquelength" means the number of unique words contained in this stream of this document. The node "termfrequency" indicates how many times of query term "termid" appears in this stream of this document. Similarly, a node "docinfo" may contain multiple child nodes of "doc", a node "doc" may contain multiple nodes of "stream", and a node"stream" may contain multiple child nodes of "term".

7. We have processed queries by stemming and removing stop words, and so here the content of queries may be a little different from the original one.

References

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-180-5. doi: http://doi.acm.org/10.1145/1102351.1102363.

Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-369-7. doi: http://doi.acm.org/10.1145/1148170.1148205.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-793-3. doi: http://doi.acm.org/10.1145/1273496.1273513.

P.A. Chirita, J. Diederich, and W. Nejdl. MailRank: using ranking for spam detection. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 373–380. ACM New York, NY, USA, 2005.

M. Collins. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, July*, pages 07–12, 2002.

K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of NIPS 2001*, 2001. URL `citeseer.ist.psu.edu/crammer01pranking.html`.

K. Dave, S. Lawrence, and D.M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM Press New York, NY, USA, 2003.

J.L. Elsas, V.R. Carvalho, and J.G. Carbonell. Fast learning of document ranking functions with the committee perceptron. *Proceedings of the international conference on Web search and web data mining*, pages 55–64, 2008.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003. ISSN 1533-7928.

Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. Query dependent ranking using k-nearest neighbor. In *SIGIR '08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2008. ACM.

John Guiver and Edward Snelson. Learning to rank with softrank and gaussian processes. In *SIGIR '08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2008. ACM Press.

Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004.

E.F. Harrington. Online Ranking/Collaborative Filtering Using the Perceptron Algorithm. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 250, 2003.

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *ICANN1999*, pages 97–102, 1999. URL `citeseer.ist.psu.edu/herbrich99support.html`.

William Hersh, Chris Buckley, T. J. Leone, and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94*, pages 192–201, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

Rong Jin, Hamed Valizadegan, and Hang Li. Ranking refinement and its application to information retrieval. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 397–406, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367552.

Guy Lebanon and John D. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 363–370, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. High accuracy retrieval with multiple nested ranker. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 437–444, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: http://doi.acm.org/10.1145/1148170.1148246.

T. Minka and S. Robertson. Selection bias in the LETOR datasets. 2008.

Lan Nie, Brian D. Davison, and Xiaoguang Qi. Topical link analysis for web search. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on*

Research and development in information retrieval, pages 91–98, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: http://doi.acm.org/10.1145/1148170.1148189.

B. Pang and L. Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics Morristown, NJ, USA, 2005.

T. Qin, T.Y. Liu, J. Xu, and H. Li. How to Make LETOR More Useful and Reliable. In *Proceedings of SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008a.

Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, Zheng Chen, and Wei-Ying Ma. A study of relevance propagation for web search. In *SIGIR '05*, pages 408–415, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-034-5. doi: http://doi.acm.org/10.1145/1076034.1076105.

Tao Qin, Tie-Yan Liu, Wei Lai, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. Ranking with multiple hyperplanes. In *SIGIR '07*, pages 279–286, New York, NY, USA, 2007a. ACM Press. ISBN 978-1-59593-597-7. doi: http://doi.acm.org/10.1145/1277741.1277791.

Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. Query-level loss functions for information retrieval. *Information Processing & Management*, 2007b.

Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, Wen-Ying Xiong, and Hang Li. Learning to rank relational objects and its application to web search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 407–416, New York, NY, USA, 2008b. ACM. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367553.

A. Shakery and C.X. Zhai. Relevance Propagation for Topic Distillation UIUC TREC-2003 Web Track Experiments. In *Proceedings of TREC*, pages 673–677, 2003.

A. Shashua and A. Levin. Taxonomy of large margin principle algorithms for ordinal regression problems. In *Proceedings of NIPS 2002*, 2002. URL `citeseer.ist.psu.edu/shashua02taxonomy.html`.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 77–86, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-927-9. doi: http://doi.acm.org/10.1145/1341531.1341544.

Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. Frank: a ranking method with fidelity loss. In *SIGIR '07*, pages 383–390, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-597-7. doi: http://doi.acm.org/10.1145/1277741.1277808.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank - theory and algorithm. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, New York, NY, USA, 2008. ACM Press.

J. Xu, Y. Cao, H. Li, and M. Zhao. Ranking definitions with supervised learning methods. In *International World Wide Web Conference*, pages 811–819. ACM Press New York, NY, USA, 2005.

Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07*, pages 391–398, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-597-7. doi: http://doi.acm.org/10.1145/1277741.1277809.

Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. Directly optimizing evaluation measures in learning to rank. In *SIGIR '08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2008a. ACM Press.

Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. Directly optimizing evaluation measures in learning to rank. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–114, New York, NY, USA, 2008b. ACM. ISBN 978-1-60558-164-4. doi: http://doi.acm.org/10.1145/1390334.1390355.

Gui-Rong Xue, Qiang Yang, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Exploiting the hierarchical structure for link analysis. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: http://doi.acm.org/10.1145/1076034.1076068.

J.Y. Yeh, J.Y. Lin, H.R. Ke, and W.P. Yang. Learning to rank for information retrieval using genetic programming. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR '07*, pages 271–278, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-597-7. doi: http://doi.acm.org/10.1145/1277741.1277790.

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to Ad Hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM Press New York, NY, USA, 2001.