ListBM: A Learning-to-Rank Method for XML Keyword Search

Ning Gao, Zhi-Hong Deng, Yong-Qing Xiang, and Yu Hang

Key Laboratory of Machine Perception (Ministry of Education)
School of Electronic Engineering and Computer Science, Peking University
{Nanacream,pkucthh}@gmail.com,
{xiangyq,zhdeng}@cis.pku.edu.cn

Abstract. This paper describes Peking University's approach to the Ad Hoc Track. In our first participation, results for all four tasks were submitted: the Best In Context, the Focused, the Relevance In Context and the Thorough. Based on retrieval method Okapi BM25, we implement two different ranking methods NormalBM25 and LearningBM25 according to different parameter settings. Specially, the parameters used in LearningBM25 are learnt by a new learning method called ListBM. The evaluation result shows that LearningBM25 is able to beat NormalBM25 in most tasks.

Keywords: XML keyword learn-to-rank.

1 Introduction

INEX Ad Hoc Track [1] aims to evaluate performance in retrieving relevant results (e.g. XML elements or documents) to a certain query. Based on lots of research and comparative experiments, Okapi BM25 [2] is confirmed to be an effective ranking method. Plus, evaluation results of Ad Hoc Track show that Okapi BM25 performs better than some other frequently cited ranking models, such as TF*IDF [3] and so on. Motivated by BM25's excellent performance, many participants prefer BM25 as their basic retrieval model. In INEX 2008 Ad Hoc Track, University of Waterloo [4] outperforms in all three tasks of Measured as Focused Retrieval, known as Best in Context, Focused and Relevance in Context. The ranking system Waterloo used is "a biased BM25 and language modeling, in addition to Okapi BM25" [4].

However, in Okapi BM25 formula, there are several parameters used to adjust the proportion of element length and term frequency (tf) in the final score and they are frequently set manually by participants. Here we note that different parameter settings might lead to totally different evaluation results. Thus, in order to get a more rigorous, evidence-based and data-based parameter setting, a listwise machine learning method to learn the parameter settings is proposed. We call it listBM.

In detail, figure 1 shows the architecture of our ranking system. Firstly, when user submits a query, a results recognizer will calculate the result elements by matching the *Keywords* and the *Inverted Index*. An element is defined as a result element only if it contains all the keywords. The output of results recognizer is a Results Set, in which result elements are disordered. Therefore, a ranking method BM25 is introduced to sort these result elements according to their relevance to the query. However, according to different parameter settings used in BM25, we implement two different ranking models NormalBM25 and LearningBM25. In NormalBM25, the parameter setting we used is same as what Waterloo used in INEX 2008. We call this parameter setting the Origin Parameter Setting. The result list ranked by using this parameter setting is called *NormalBM25 Results*. While in LearningBM25 model, the parameters are learned by a machine learning method called ListBM, to be introduced in section 3. The result list ranked by BM25 using this *Learnt Parameter Setting* is defined as *LearningBM25 Results*. We submit both the *NormalBM25 results* and the *LearningBM25 results* in four tasks of INEX 2009 Ad Hoc Track. The evaluation results show that *LearningBM25 results* perform better than *NormalBM25 results*, indicating that our learning method ListBM indeed help to improve the performance.

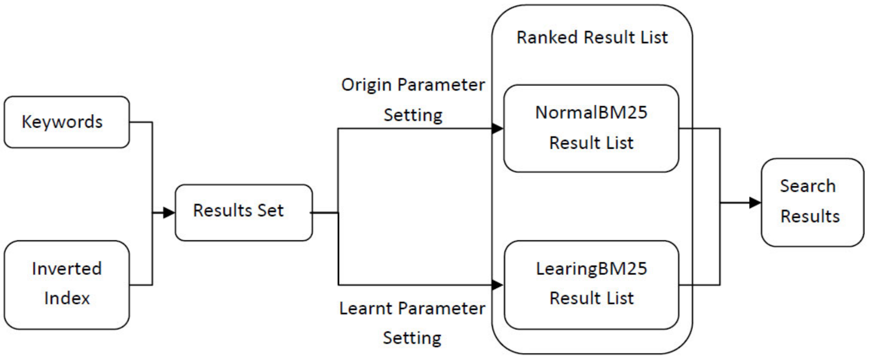


Fig. 1. Architecture of Ranking System

In section 2, we introduce the concepts of BM25 and background of machine learning method we used. Section 3 describes our learning method ListBM. In section 4, we show the evaluation results. Section 5 is the conclusion and future work.

2 Related Work

2.1 Okapi BM25[2]

BM25 is a widely quoted ranking method. It shows excellent performance referring to the evaluation results of INEX in the past years. For our method, to score an element according to its relevance to a certain query, we chose BM25 as our basic ranking model. In detail, the score is defined as follows.

$$\text{score}(e,Q)=\sum_{t\in Q} W_t \cdot \frac{(k_1+1)\cdot tf(t,e)}{k_1\cdot(1-b+b\cdot\frac{len(e)}{avel})+tf(t,e)} \tag{1}$$

Score(e,Q) measures the relevance of element e to a certain query Q; W_t is the weight of term t indicating the inverse term frequency(IDF) of term t in collection. tf(t,e) is the frequency of term t appearing in element e; len(e) denotes the length of element e and avel denotes the average length of elements in whole collection. Two parameters, $k_1$and b, are used to balance the weight of term frequency (tf) and element length(len) in final score.

2.2 Learning-to-Rank Methods

Learning-to-rank methods focus on using machine learning algorithms for better ranking. Many learning-to-rank algorithms have been proposed. According to different "instance" they use, learning-to-rank methods can be classified into three categories [5]: pointwise, pairwise and listwise.

In pointwise methods, documents are used as learning instance. The relevant score of a document is calculated by its features such as term frequency (tf), tag name and document links. This kind of algorithm attempts to find classification engine that can mark document as relevant or irrelevant correctly.

Pairwise methods, such as RankBoost [6] and RankSVM [7], take document pair as learning instance. Consider two documents d_1 and d_2, if d_1 is more relevant than d_2 to a certain query Q, then the document pair (d_1,d_2) is set to 1, otherwise it is set to -1. Pairwise methods target at training a learning engine to find the best document pair preferences.

In listwise methods, document list is taken as learning instance to train ranking engines. To find the best ranked list is the final goal. There are several well-known listwise methods such as Listnet [5], ListMLE [8], SVM-MAP[9] and so on.

Comparative tests [10] have shown that listwise methods perform best in these three categories.

3 Learning-to-Rank Method ListBM

3.1 ListBM

In NormalBM model, the parameter setting used in BM25 is same as Waterloo set in INEX 2008. In LearningBM model, the parameters are learnt by a learning method called ListBM. We use INEX 2008 data collection as the training data base.

In training, there is a set of query $Q=\{q^1,q^2,\ldots,q^m\}$. Each query q^i is associated with a ranked list of relevant documents $D^i= (d^i_1, d^i_2, \dots , d^i_n)$, where d^i_j denotes the j-th relevant document to query q^i. These relevant documents lists, downloaded from the website of INEX, are used as standard results in our training. What's more, for each query q^i, we use basic ranking method BM25 mentioned in 2.1 to get a list of search results RL^i. Results returned by BM25 are all in form of elements. The first n result elements of RL^i are recorded in $R^i=(r^i_1\ r^i_2,\ldots, r^i_n)$. Then each documents list D^i and elements list R^i form a "instance".

The loss function is defined as the "distance" between standard results lists D^i and search results lists R^i. Therefore, the objective of learning is formalized as minimization of the total losses with respect to the training data.

$$\sum_{i=1}^{m} L(D^i, R^i) \tag{2}$$

Suppose there are two search results R, R' and a standard result D, the definition of loss function should meet the following two criterions:

- The loss value should be inversely proportional to the recall. If R contains more relevant results appeared in D than R' does, then the loss value of R should be smaller than the loss value of R'.
- The loss value should be inversely proportional to the precision. If the relevant content contained by R has the higher relevance degree (they appear in the top-ranking documents in D), then the loss value of R should be lower.

According to these two criterions, we define the loss function for a single query q^i as:

$$L(D^i, R^i) = \frac{mn^2}{\sum_{j=1}^{n} \mathrm{rank}_j^i} \tag{3}$$

$$\mathrm{rank}_j^i = \begin{cases} 0, & (\text{if the j-th result in } R^i \text{ doesn' appear in } D^i) \\ k, & (\text{if the j-th result in } R^i \text{ is contained by the k-th result in } D^i) \end{cases} \tag{4}$$

Where m is the number of queries in Q and n is the number of relevant documents in D^i corresponding to a certain query q^i. rank_j^i is divided into two parts: (1) if the j-th result element in R^i isn't contained by any relevant documents in D^i, then rank_j^i is set to 0; (2) if the j-th result element in R^i is contained by the k-th relevant document d_k^i in D^i, then rank_j^i is set to k. Apparently, an element can only be contain by one document. Since the documents in D are different, it is impossible that there are more than one documents in D contains the j-th result in R.

Using the pair of standard results D and searching results R as "instance", L defined above as loss function, we implement a learning method ListBM to learn the two parameters k_1 and b in BM25 separately. Algorithm 1 describes the procedure of learning parameter k_1.

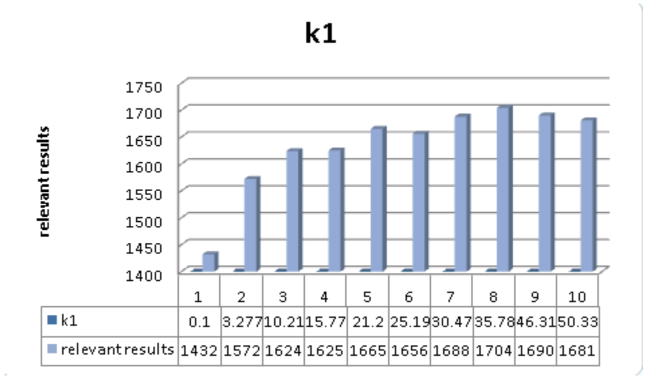
Algorithm 1. ListBM: The process of learning k_1

```
Input: query Q, relevant documents D
Parameter: $k_1$, number of iterations T,
Initialize: b=0.8, $\sum_{i=1}^{m} L(D^i, R^i)=+\infty$
for t=1 to T do
    for i=1 to m do
        search the $q^i$ using BM25, get $R^i$;
        compute $L(D^i, R^i)$
        update $k_1 += \frac{1}{L(D^i,R^i)}$
    end for
    if $\sum_{i=1}^{m} L(D^i, R^i)$ increases, then break;
end for
output the pair of {$k_{min}$, min $\sum_{i=1}^{m} L(D^i, R^i)$; }
```

When learning k_1, parameter b will be initialized to 0.8. While randomly entering an original value of k_1, the loop won't end until the new updated k_1 begin to increase the loss value. The output is a pair of $\{k_{min}$, $\min \sum_{i=1}^{m} L(D^i, R^i)$; }, in which $\min \sum_{i=1}^{m} L(D^i, R^i)$ is the minimum loss value got in the process and k_{min} is the corresponded k_1. While learning parameter b, k_1 is a content value 4 and b is updated according to the loss function. What's more, the output will be the pair of $\{b_{min}$, $\min \sum_{i=1}^{m} L(D^i, R^i)$; }, in which b_{min} is the better b leading to minimum loss value.

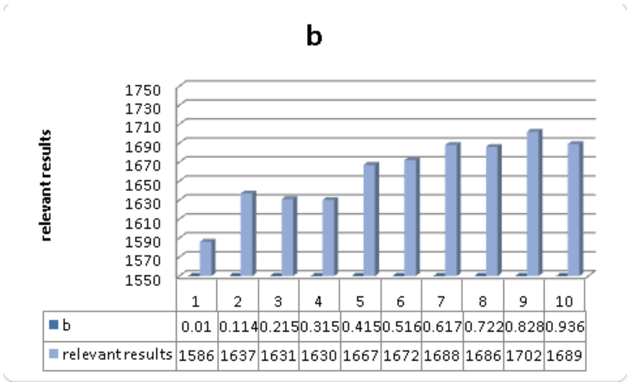
3.2 Experiments

We implemented our ranking system in C++. The data collection we use is the English files of wiki provided by INEX 2008 Ad Hoc Track. The total size with these 659,388 files is 4.6G. In the process of reading and analyzing these XML files, we remove all the stop word from a standard stop word list before stemming. We use 8 queries from INEX 2008 topic pool in the training process. Totally 4800 documents are signed as relevant results. Figure 2 and figure 3 show the learning results of k_1 and b.



	1	2	3	4	5	6	7	8	9	10
■ k1	0.1	3.277	10.21	15.77	21.2	25.19	30.47	35.78	46.31	50.33
■ relevantresults	1432	1572	1624	1625	1665	1656	1688	1704	1690	1681

Fig. 2. Learning result of k_1



	1	2	3	4	5	6	7	8	9	10
■ b	0.01	0.114	0.215	0.315	0.415	0.516	0.617	0.722	0.828	0.936
■ relevantresults	1586	1637	1631	1630	1667	1672	1688	1686	1702	1689

Fig. 3. Learning result of b

Figure 2 illustrates the learning results of k_1. In NormalBM model, the k_1 is set to 4. As is shown, in total 4800 search results, only 1572 results are relevant according to the standard results when k_1 is set to 3.277. The best set of k_1 is 35 so that the relevant results can reach up to 1704.

Figure 3 shows the learning result of b. The origin set of b is 0.8 according to the parameter setting of Waterloo. Result shows that b=0.8 is indeed the best set leading to a better searching performance. Hence, the parameter setting of {k_1, b} is set to {35, 0.8} in LearningBM model and {4, 0.8} in NormalBM model.

4 Evaluation Results

We submit both NormalBM results and LearningBM results for all four tasks to compare the performance. The original results are taken as the results of Thorough task. For Best in Context task, we return the most relevant element with the highest score as the best entry of a document. Moreover, based on the original result, for Focused task, we remove the elements which are contained by other element. We get evaluation results for three of four tasks. Table 1 shows the evaluation results of Measured as Focused Retrieval in which the search results are elements. Table 2 describes the search performance of Measured as Document Retrieval tasks. We can say that, in most conditions, the retrieval effectiveness of LearningBM is better than that of NormalBM.

Table 1. Evaluation results of element retrieval

	Best in Context	Focused	Thorough
LearningBM25	0.0953	0.3072	0.0577
NormalBM25	0.0671	0.1779	0.0521

Table 2. Evaluation results of document retrieval

	Best in Context	Focused	Thorough
LearningBM25	0.2382	0.2382	0.1797
NormalBM25	0.1849	0.1847	0.1689

5 Conclusion and Future Work

We propose a new learning method ListBM to learn the parameters in ranking method BM25. ListBM is a listwise learning method, using the data source of INEX 2008 Ad Hoc Track as training data base. The evaluation results show that parameter setting learnt by ListBM performs better than parameter setting set manually.

For the future work, we will continue to work on the following problems:

- The training data we used is the collection of INEX 2008 Ad Hoc Track. However, data collection has changed a lot for INEX 2009. We will study the learning to rank method on the new collection in the future.

- There are only 8 queries used in training. For the further study, more queries from INEX 2009 topics pool will be searched in the learning process.
- We will propose new definition of "distance" between the search result list and the standard result list. Furthermore, a more reasonable loss function and a new updating method will be introduced.

Acknowledgments

This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2009AA01Z136 and the National Natural Science Foundation of China under Grant No. 90812001.

References

[1] `http://www.inex.otago.ac.nz/`
[2] Theobald, M., Schenkel, R., Wiekum, G.: An Efficient and Versatile Query Engine for TopX Search. In: VLDB, pp. 625–636 (2005)
[3] Carmel, D., Maarek, Y.S., Mandelbrod, M., et al.: Searching XML documents via XML fragments. In: SIGIR, pp. 151–158 (2003)
[4] Itakura, K.Y., Clarke, C.L.A.: University of Waterloo at INEX 2008: Adhoc, Book, and Link-the-Wiki Tracks. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 116–122. Springer, Heidelberg (2009)
[5] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to Rank: From Pairwise Approach to Listwise Approach. Microsoft technique report
[6] Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res., 933–969 (2003)
[7] Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: Proceedings of ICANN, pp. 97–102 (1999)
[8] Xia, F., Liu, T.-Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: ICML 2008: Proceedings of the 25th international conference on Machine learning, pp. 1192–1199 (2008)
[9] Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A Support Vector Method for Optimizing Average Precision. In: Proceedings of the ACM Conference on Research and Development in Information Retrieval, SIGIR (2007)
[10] Zhang, M., Kuang, D., Hua, G., Liu, Y., Ma, S.: Is learning to rank effective for Web search? In: SIGIR 2009 workshop (2009)