ListOPT: Learning to Optimize for XML Ranking

Ning Gao[1], Zhi-Hong Deng[1,2,⋆], Hang Yu[1], and Jia-Jian Jiang[1]

[1]Key Laboratory of Machine Perception (Ministry of Education),
School of Electronic Engineering and Computer Science, Peking University
[2]The State Key Lab of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China
ninggao@pku.edu.cn, zhdeng@cis.pku.edu.cn, pkucthh@gmail.com,
jjjoke@163.com

Abstract. Many machine learning classification technologies such as boosting, support vector machine or neural networks have been applied to the ranking problem in information retrieval. However, since the purpose of these learning-to-rank methods is to directly acquire the sorted results based on the features of documents, they are unable to combine and utilize the existing ranking methods proven to be effective such as BM25 and PageRank. To solve this defect, we conducted a study on learning-to-optimize, which is to construct a learning model or method for optimizing the free parameters in ranking functions. This paper proposes a listwise learning-to-optimize process ListOPT and introduces three alternative differentiable query-level loss functions. The experimental results on the XML dataset of Wikipedia English show that these approaches can be successfully applied to tuning the parameters used in an existing highly cited ranking function BM25. Furthermore, we found that the formulas with optimized parameters indeed improve the effectiveness compared with the original ones.

Keywords: learning-to-optimize, ranking, BM25, XML.

1 Introduction

Search engines have become an indispensable part of life and one of the key issues on search engine is ranking. Given a query, the ranking modules can sort the retrieval documents for maximally satisfying the user's needs. Traditional ranking methods aim to compute the relevance of a document to a query, according to the factors, term frequencies and links for example. The search result is a ranked list in which the documents are sequenced by their relevance score in descending order. These kinds of methods include the content based functions such as TF*IDF [1] and BM25 [2], and link based functions such as PageRank [3] and HITS [4].

Recently, machine learning technologies have been successfully applied to information retrieval, known and named as "learning-to-rank". The main procedure of "learning-to-rank" is as follow: In learning module, a set of queries is given, and each of the queries is associated with a ground-truth ranking list of documents. The process targets

⋆ Corresponding author.

at creating a ranking model that can precisely predict the order of documents in the ground-truth list. Many learning-to-rank approaches have been proposed and based on the differences of their learning samples, these methods can be classified into three categories [5]: pointwise, pairwise and listwise. Taking single document as learning object, the pointwise based methods intent to compute the relevance score of each document with respect to their closeness to the ground-truth. On the other side, pairwise based approaches take the document pair as learning sample, and rephrase the learning problem as classification problem. Lisewise based approaches take a ranked list as learning sample, and measure the differences between the current result list and the ground-truth list via using a loss function. The learning purpose of listwise methods is to minimize the loss. The experimental results in [5] [11] [12] show that the listwise based methods perform the best among these three kinds of methods.

It is worth noting that, from the perspective of ranking, the aforementioned learning-to-rank methods belong to the learning based ranking technologies. Here the search results are directly obtained from the learning module, without considering the traditional content based or link based ranking functions. However, there is no evidence to confirm that the learning based methods perform better than all the other classic content based or link based methods. Accordingly, to substitute the other two kinds of ranking technologies with the learning based methods might not be appropriate.

We hence consider a learning-to-optimize method ListOPT that can combine and utilize the benefits of learning-to-rank methods and traditional content based methods. Here the ranking method is the extension to the widely known ranking function BM25. Due to previous studies, experiments are conducted on selecting the parameters of BM25 with the best performance, typically after thousands of runs. However, this simple but exhaustive procedure is only applicable to the functions with few free parameters. Besides, whether the best parameter values are in the testing set is also under suspect. To attack this defect, a listwise learning method to optimize the free parameters is introduced.

Same as learning-to-rank methods, the key issue of learning-to-optimize method is the definition of loss function. In this paper, we discuss the effect of three distinct definition of loss in the learning process and the experiments show that all three loss functions converge. The experiments also reveal that the ranking function using tuned parameter set indeed performs better.

The primary contributions of this paper include: (1) proposed a learning-to-optimize method which combine and utilize the traditional ranking function BM25 and listwise learning-to-rank method, (2) introduced the definition of three query-level loss functions on the basis of cosine similarity, Euclidean distance and cross entropy, confirmed to converge by experiments, (3) the verified the effectiveness of the learning-to-optimize approach on a large XML dataset Wikipedia English[6].

The paper is organized as follows. In section 2, we introduce the related work. Section 3 gives the general description on learning-to-optimize approach ListOPT. The definition of the three loss functions are discussed in section 4. Section 5 reports our experimental results. Section 6 is the conclusion and future work.

2 Related Work

2.1 Learning-to-Rank

In recent years, many machine learning methods were applied to the problem of ranking for information retrieval. The existing learning-to-rank methods fall into three categories, pointwise, pairwise and listwise. The pointwise approaches [7] are firstly proposed, transforming the ranking problem into regression or classification on single candidate documents. On the other side, pairwise approaches, published later, regard the ranking process as a classification of document pairs. For example, given a query Q and an arbitrary document pair $P = (d_1, d_2)$ in the data collection, where d_i means the i-th candidate document, if d_1 shows higher relevance than d_2, then object pair P is set as $\Phi(p) > 0$, otherwise P is marked as $\Phi(p) < 0$. The advantage of pointwise and pairwise approaches is that the existing classification or regression theories can be directly applied. For instance, borrowing support vector machine, boosting and neural network as the classification model leads to the methods of Ranking SVM [8], RankBoost [9] and RankNet [10].

However, the objective of pointwise and pairwise learning methods is to minimize errors in classification of single document or document pairs rather than to minimize errors in ranking of documents. To overcome this drawback of the aforementioned two approaches, listwise methods, such as ListNet [5], RankCosine [11] and ListMLE [12], are proposed. In lisewise approaches, the learning object is the result list and various kinds of loss functions are defined to measure the similarity of the predict result list and the ground-truth result list. ListNet, the first listwise approach proposed by Cao et al., uses the cross entropy as loss function. Qin et al. discussed about another listwise method called RankCosine, where the cosine similarity is defined as loss function. Xia et al. introduced likelihood loss as loss function in the listwise learning-to-rank method ListMLE.

2.2 Ranking Function BM25

In information retrieval, BM25 is a highly cited ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s. Though BM25 is proposed to rank the HTML format documents originally, it was introduced to the area of XML documents ranking in recent years. In the last three years of INEX[1] [6] Ad Hoc track [2] [17] [18] [19], all the search engines that perform the best use BM25 as basic ranking function. To improve the performance of BM25, Taylor et al. introduced the pairwise learning-to-rank method RankNet to tune the parameters in BM25, named as *RankNet Tuning* method [13] in this paper. However, as mentioned in 2.1, the inherent disadvantages of pairwise methods had a pernicious influence on the

[1] Initiative for the Evaluation of XML retrieval (INEX), a global evaluation platform, is launched in 2002 for organizations from Information Retrieval, Database and other relative research fields to compare the effectiveness and efficiency of their XML search engines.

[2] In Ad Hoc track, participates are organized to compare the retrieval effectiveness of their XML search engines.

approach. Experiments in section 5 will compare the effectiveness of *RankNet Tuning* with the other methods proposed in this paper.

3 ListOPT: A Learning-to-Optimize Approach

In this section, we describe the details in the learning-to-optimize approach. Firstly, we give the formal definition of the ranking function BM25 used in XML retrieval and analyze the parameters in the formula. Then, the training process of the listwise learning-to-optimize approach ListOPT is proposed in 3.2.

3.1 BM25 in XML Retrieval

Unlike the HTML retrieval, the searching retrieval results are elements in XML retrieval, the definition of BM25 is thus different from the traditional BM25 formula used in HTML ranking. The formal definition is as follow:

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

$$W_t = log \frac{Nd}{n(t)}$$

In the formula, $tf(t, e)$ is the frequency of keyword t appeared in element e; Nd is the number of files in the collection; $n(t)$ is the number of files that contains keyword t; $len(e)$ is the length of element e; $avel$ is average length of elements in the collection; Q is a set of keywords; $ps(e, Q)$ is the predict relevance score of element e corresponding to query Q; b and k are two free parameters.

As observed, the parameters in BM25 fall into three categories: *constant parameters*, *fixed parameters* and *free parameters*. For example, parameters describing the features of data collection like *avel* and *Nd* are defined as *constant parameters*. Given a certain query and a candidate element, $tf(t, e)$ and $len(e)$ in the formula are fixed values. This kind of parameters is called *fixed parameters*. Moreover, *free parameters*, such as k and b in the function, are set to make the formula more adaptable to various kinds of data collections. Therefore, the ultimate objective of learning-to-optimize approach is to learn the optimal set of *free parameters*.

3.2 Training Process

In training, there is a set of query $Q = \{q^1, q^2, , q^m\}$. Each query q^i is associated with a list of candidate elements $E^i = (e^i_1, e^i_2, , e^i_{n(i)})$, where e^i_j denotes the the j-th candidate element to query q^i and $n(i)$ is the size of E^i . The candidate elements are defined as the elements that contain at least one occurrence of each keyword in the query. Moreover, each candidate elements list E^i is associated with a ground-truth list $G^i = (g^i_1, g^i_2, , g^i_{n(i)})$, indicating the relevance score of each elements in E^i. Given that the data collection we used only contains information of whether or not the passages in a document are relevant, we apply the *F measure* cite14 to evaluate the ground truth score. Given a query q^i, the ground-truth score of the j-th candidate element is defined as follow:

$$precision = \frac{relevant}{relevant + irrelevant}$$
$$recall = \frac{relevant}{REL}$$
$$g^i_j = \frac{(1 + 0.1^2) \cdot precision \cdot recall}{0.1^2 \cdot precision + recall} \quad (2)$$

In the formula, *relevant* is the length of relevant contents highlighted by user in e, while *irrelevant* stands for the length of irrelevant parts. *REL* indicates the total length of relevant contents in the data collection. The general bias parameter β is set as 0.1, denoting that the weight of precision is ten times as much as recall.

Furthermore, for each query q^i, we use the ranking function BM25 mentioned in 3.1 to get the predict relevant score of each candidate element, recorded in $R^i = (r^i_1 r^i_2, , r^i_{n(i)})$. Then each ground-truth score list G^i and predicted score 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.

$$\sum_{i=1}^{m} L(G^i, R^i) \quad (3)$$

In each training epoch, the ranking function BM25 was used to compute the predicted score R^i. Then the learning module replaced the current free parameters with the new parameters tuned according to the loss between G^i and R^i. Finally the process stops either while reaching the limit cycle index or when the parameters do not change.

4 Loss Functions

In this section, three query level loss functions and the corresponding tuning formulas are discussed. Here the three definitions of loss are based on cosine similarity, Euclidean distance and cross entropy respectively. After computing the loss between the ground-truth G^i and the predicted R^i, the two free parameters k and b in BM25 are tuned as formula (4). Especially, and are set to control the learning speed.

$$k \leftarrow k - \eta \cdot \triangle k$$
$$b \leftarrow b - \lambda \cdot \triangle b \quad (4)$$

4.1 Cosine Similarity

Widely used in text mining and information retrieval, cosine similarity is a measure of similarity between two vectors by finding the cosine of the angle between them. The definition of the query level loss function based on cosine similarity is:

$$L(G^i, R^i) = \frac{1}{2} \cdot (1 - \frac{\sum_{i=1}^{n(i)} \Psi^i_j \cdot g^i_j \cdot r^i_j}{\sqrt{\sum_{j=1}^{n(i)} (g^i_j)^2} \sqrt{\sum_{j=1}^{n(i)} (r^i_j)^2}}) \quad (5)$$

Note that in large data collection, given a query, the amount of relevant documents is regularly much less than the number of irrelevant documents. So that a penalty function Ψ_j^i is set to avoid the learning bias on irrelevant documents. Formula (6) is the weight of relevant documents in learning procedure, while formula (7) is the weight of irrelevant document. The formal definition is as follow:

$$\Psi_j^i = \begin{cases} \dfrac{NR^i + NIR^i}{NR^i}, & \text{if } (g_j^i > 0) \qquad (6) \\ \dfrac{NR^i + NIR^i}{NIR^i}, & \text{if } (g_j^i = 0) \qquad (7) \end{cases}$$

Where NR^i is the number of relevant elements according to query q^i and NIR^i is the number of irrelevant ones. After measuring the loss between the ground-truth results and the predicted results, the adjustment range parameters $\triangle k$ and $\triangle b$ are determined according to the derivatives of k and b:

With respect to k:

$$\triangle k = \sum_{q=1}^{m} \frac{\partial L(G^i, R^i)}{\partial k}$$

$$= (-\frac{1}{2}) \cdot \sum_{q=1}^{m} \Psi_j^q \cdot (\frac{\sum_{j=1}^{n(i)} g_j^q \cdot \frac{\partial r_j^q}{\partial k}}{\sqrt{\sum_{j=1}^{n(i)} (r_j^q)^2} \cdot \sqrt{\sum_{j=1}^{n(i)} (g_j^q)^2}} - \frac{(\sum_{j=1}^{n(i)} r_j^q \cdot g_j^q)(\sqrt{\sum_{j=1}^{n(i)} (g_j^q)^2} \cdot \frac{{}_{j=1}^{n(i)} r_j^q \cdot \frac{\partial r_j^q}{\partial k}}{{}_{j=1}^{n(i)} (r_j^q)^2})}{(\sqrt{\sum_{j=1}^{n(i)} (r_j^q)^2} \sqrt{\sum_{j=1}^{n(i)} (g_j^q)^2})^2}) \qquad (8)$$

In which:

$$\frac{\partial r_j^q}{\partial k} = \sum_{t \in Q} W_t \cdot \frac{tf(t,e) \cdot (tf(t,e) + k \cdot (1 - b + b \cdot \frac{len(e)}{avel})) - tf(t,e) \cdot (k+1) \cdot (1 - b + b \cdot \frac{len(e)}{avel})}{(tf(t,e) + k \cdot (1 - b + b \cdot \frac{len(e)}{avel}))^2} \qquad (9)$$

b analogously:

$$\triangle b = \sum_{q=1}^{m} \frac{\partial L(G^i, R^i)}{\partial b}$$

$$= (-\frac{1}{2}) \cdot \sum_{q=1}^{m} \Psi_j^q \cdot (\frac{\sum_{j=1}^{n(i)} g_j^q \cdot \frac{\partial r_j^q}{\partial b}}{\sqrt{\sum_{j=1}^{n(i)} (r_j^q)^2} \cdot \sqrt{\sum_{j=1}^{n(i)} (g_j^q)^2}} - \frac{(\sum_{j=1}^{n(i)} r_j^q \cdot g_j^q)(\sqrt{\sum_{j=1}^{n(i)} (g_j^q)^2} \cdot \frac{{}_{j=1}^{n(i)} r_j^q \cdot \frac{\partial r_j^q}{\partial b}}{{}_{j=1}^{n(i)} (r_j^q)^2})}{(\sqrt{\sum_{j=1}^{n(i)} (r_j^q)^2} \sqrt{\sum_{j=1}^{n(i)} (g_j^q)^2})^2}) \qquad (10)$$

In which:

$$\frac{\partial r_j^q}{\partial b} = \sum_{t \in Q} W_t \cdot \frac{tf(t,e) \cdot (k+1) \cdot k \cdot (1 - \frac{len}{avel})}{(tf(t,e) + k \cdot (1 - b + b \cdot \frac{len(e)}{avel}))^2} \qquad (11)$$

4.2 Euclidean Distance

The Euclidean distance is also used in the definition of loss function. The circumscription of penalty parameter Ψ_j^i is the same as in formula (6) and (7). Hence, the loss function based on Euclidean distance is defined in formula (12).

$$L(G^i, R^i) = \sqrt{\sum_{j=1}^{n(i)} (\Psi_j^i)^2 \cdot (r_j^i - g_j^i)^2} \quad (12)$$

The same as cosine similarity loss, we derive the derivatives of the loss function based on Euclidean distance with respect to k and b. The definition of $\frac{\partial r_j^q}{\partial k}$ and $\frac{\partial r_j^q}{\partial b}$ are the same as in formula (9) and formula (11) respectively.

With respect to k:

$$\triangle k = \sum_{q=1}^{m} \frac{\partial L(G^i, R^i)}{\partial k} = \sum_{q=1}^{m} \frac{\sum_{j=1}^{n(i)} (\Psi_j^i)^2 (r_j^q - g_j^q) \cdot \frac{\partial r_j^q}{\partial k}}{\sqrt{\sum_{j=1}^{n(i)} (\Psi_j^i)^2 (r_j^q - g_j^q)^2}} \quad (13)$$

b analogously:

$$\triangle b = \sum_{q=1}^{m} \frac{\partial L(G^i, R^i)}{\partial b} = \sum_{q=1}^{m} \frac{\sum_{j=1}^{n(i)} (\Psi_j^i)^2 (r_j^q - g_j^q) \cdot \frac{\partial r_j^q}{\partial b}}{\sqrt{\sum_{j=1}^{n(i)} (\Psi_j^i)^2 (r_j^q - g_j^q)^2}} \quad (14)$$

4.3 Cross Entropy

$$L(G^i, R^i) = -\sum_{j=1}^{n(i)} \Psi_j^i \cdot r_j^i \cdot log(g_j^i) \quad (15)$$

When considering cross entropy as metric, the loss function turns to formula (15). Moreover, the penalty parameter Ψ_j^i in the formula is the same as in formula (6) and (7) and the detailed tuning deflection of $\triangle k$ and $\triangle b$ is defined in formula (16) and formula (17) respectively. Additionally, the definition of $\frac{\partial r_j^q}{\partial k}$ and $\frac{\partial r_j^q}{\partial b}$ are the same as in formula (9) and formula (11). With respect to k:

$$\triangle k = \sum_{q=1}^{m} \frac{\partial L(G^i, R^i)}{\partial k} = \sum_{q=1}^{m} \Psi_j^q \cdot \Big(-\sum_{j=1}^{n(i)} r_j^q \cdot \frac{\partial r_j^q}{\partial k} + \frac{1}{\sum_{j=1}^{n(i)} g_j^q} \cdot \sum_{j=1}^{n(i)} g_j^q \cdot \frac{\partial r_j^q}{\partial k}\Big) \quad (16)$$

b analogously:

$$\triangle b = \sum_{q=1}^{m} \frac{\partial L(G^i, R^i)}{\partial b} = \sum_{q=1}^{m} \Psi_j^q \cdot \Big(-\sum_{j=1}^{n(i)} r_j^q \cdot \frac{\partial r_j^q}{\partial b} + \frac{1}{\sum_{j=1}^{n(i)} g_j^q} \cdot \sum_{j=1}^{n(i)} g_j^q \cdot \frac{\partial r_j^q}{\partial b}\Big) \quad (17)$$

5 Experiment

In this section, the XML data set used in comparison experiments is first introduced. Then in section 5.2 we compare the effectiveness of the optimized ranking function BM25 under two evaluation criterions: MAP [15] and NDCG [16]. Additionally in section 5.3, we focus on testing the association between the number of training queries and the optimizing performance under the criterion of MAP.

5.1 Data Collection

The data collection used in the experiments consists of 2,666,190 English XML files from Wikipedia, used by INEX Ad Hoc Track. The total size of these files is 50.7GB. The query set consists of 68 different topics from competition topics of Ad Hoc track, in which 40 queries are considered as training queries and others are test queries. Each query in the evaluation system is bound to a standard set of highlighted ”relevant content”, which is recognized manually by the participants of INEX. In the experiments, the training regards these highlighted ”relevant content” as ground truth results.

5.2 Effect of BM25 Tuning

To explore the effect of the learning-to-optimize method ListOPT, we evaluate the effectiveness of different parameter sets. In the comparison experiments, Traditional Set stands for a highly used traditional set: $k = 2, b = 0.75$; *RankNet Tuning* stands for the tuning method proposed in [10]; *cosine similarity, Euclidean distance* and *cross entropy* are the learning-to-optimize methods using cosine similarity, Euclidean distance and cross entropy as loss function respectively.

Evaluation System of Ad Hoc is the standard experiment platform in effectiveness comparison here. We evaluate the searching effectiveness of the aforementioned five methods in two criterions: MAP and NDCG. In MAP evaluation, we choose interpolated precision at 1% recall (iP[0.01]), precision at 10% recall (iP[0.10]) and MAiP as the evaluation criterions. While in NDCG evaluation, we test the retrieval effect on NDCG@1 to NDCG@ 10.

Figure 1 illustrates the comparison results under MAP measure. As is shown, the three learning-to-optimize methods proposed in this paper perform the best. It might looks confusing since that INEX 2009 Ad Hoc Focused track used to have search engine reaching ip[0.01] = 0.63, compared to 0.34 the highest in our plot. However, this effectiveness is regularly obtained by combining various ranking technologies together, such as two-layer strategy, re-ranking strategy and title tag bias, but not only BM25 itself. Given that our study is focused on BM25, it is therefore pointless to do such a combination. In this condition, the searching effectiveness scores (ip[0.01]) in this experiment are unable to show the high level as the competitive engines in INEX did.

The result presented in figure 2 show that the learning-to-optimize methods are indeed more robust in ranking tasks. The performance of the ranking methods becomes better when more results are returned. From the perspective of users, this phenomenon could be explained by the fact that INEX queries are all *information query*, meaning that the user’s purpose is to find more relevant information. On contrary, if the query is a *navigation query*, the user only needs the exact webpage. The first result is hence of highest importance and the evaluation score might decreases accordingly.

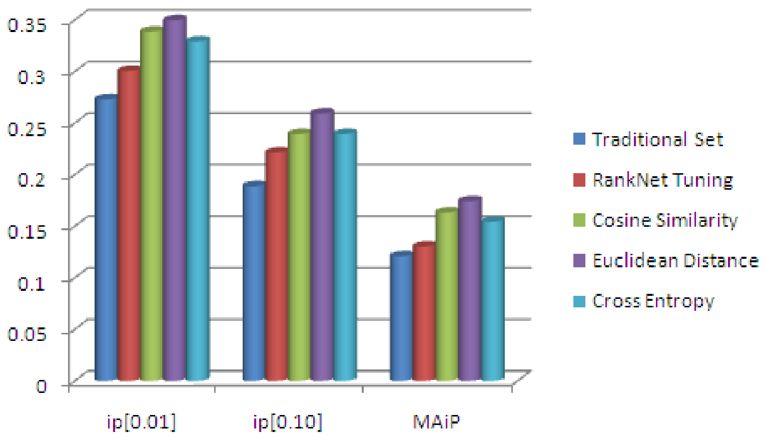

Fig. 1. Effective Comparison on MAiP

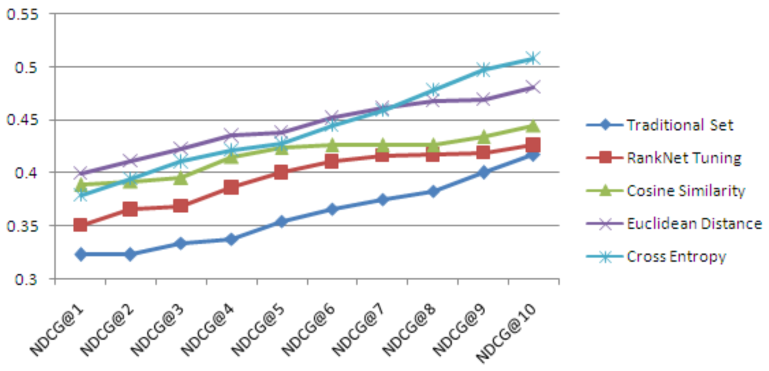

Fig. 2. Effectiveness Comparison on NDCG

5.3 Number of Training Queries

Figure 3 shows the relationship between the tuning effectiveness and the quantity of training queries. In this experiment, the number of training queries changes from 1 to 40. As illustrated, the MAiP score lines all share an obvious ascent during the first several query numbers. After that, the pulsation of the performance keeps in a low level. This situation corresponds with the learning theory: when there are few queries in the training set, the learning is overfitting. With the increasing of query samples, the performance of learning procedure gets better and better till finally the most proper parameters for the data collection are found.

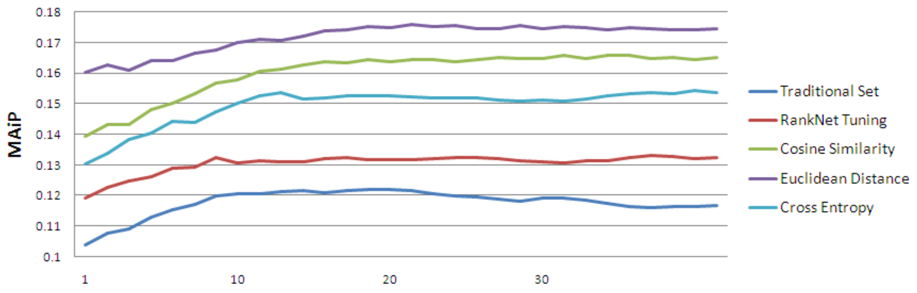

Fig. 3. Number of Training Queries

6 Conclusions and Future Work

In this paper, we proposed a learning-to-optimize method ListOPT. ListOPT combines and utilizes the benefits of the listwise learn-to-rank technology and the traditional ranking function BM25. In the process of learning, three query level loss functions based on cosine similarity, Euclidean distance and cross entropy respectively are introduced. The experiments on a XML data set Wikipedia English confirm that the learning-to-optimize method indeed leads to a better parameter set.

As future work, we will firstly try to tune W_t in the formula. Then we would like to extend the learning-to-optimize method ListOPT approach to other tuning fields, like tuning the parameters in other ranking functions or ranking methods in the future. In a further, the comparison of ListOPT and some other learning and ranking methods, such as ListNet, XRank, XReal and so on, will be done on benchmark data sets.

Acknowledgement

This work was supported by the National High-Tech Research and Development Plan of China under Grant No.2009AA01Z136.

References

1. Carmel, D., Maarek, Y.S., Mandelbrod, M., et al.: Searching XML documents via XML fragments. In: SIGIR, pp. 151–158 (2003)
2. Theobald, M., Schenkel, R., Wiekum, G.: An Efficient and Versatile Query Engine for TopX Search. In: VLDB, pp. 625–636 (2005)
3. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University (1998)
4. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. JACM, 604–632 (1998)
5. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: ICML, pp. 129–136 (2007)
6. INEX, `http://www.inex.otago.ac.nz/`
7. Nallapati, R.: Discriminative models for information retrieval. In: SIGIR, pp. 64–71 (2004)

8. Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., Hon, H.: Adapting ranking SVM to document retrieval. In: SIGIR, pp. 186–193 (2006)
9. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. JMLR, 933–969 (2003)
10. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to Rank using Gradient Descent. In: ICML, pp. 89–96 (2005)
11. Qin, T., Zhang, X.D., Tsai, M.F., Wang, D.S., Liu, T.Y., Li, H.: Query-level loss functions for information retrieval. Information Processing and Management, 838–855 (2007)
12. Xia, F., Liu, T.Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: ICML, pp. 1192–1199 (2008)
13. Taylor, M., Zaragoza, H., Craswell, N., Robertson, S., Burges, C.: Optimisation Methods for Ranking Functions with Multiple Parameters. In: CIKM, pp. 585–593 (2006)
14. van Rijsbergen, C.J.: Information Retrieval. Butterworths, London (1979)
15. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval (1999)
16. Jarvelin, K., Kekalainen, J.: IR evaluation methods for retrieving highly relevant documents. In: SIGIR, pp. 41–48 (2000)
17. Geva, S., Kamps, J., Lethonen, M., Schenkel, R., Thom, J.A., Trotman, A.: Overview of the INEX 2009 Ad Hoc Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 4–25. Springer, Heidelberg (2010)
18. 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. 132–139. Springer, Heidelberg (2009)
19. Liu, J., Lin, H., Han, B.: Study on Reranking XML Retrieval Elements Based on Combining Strategy and Topics Categorization. In: INEX, pp. 170–176 (2007)