Query Dependent Ranking for Information Retrieval by Using Query-Dependent Loss Function

Pwint Hay Mar Lwin, NangSai Moon Kham
University of Computer Studies, Yangon
pwinthaymarlwin.phml@gmail.com,moonkhamucsy@gmail.com

Abstract

Ranking is the central problem for information retrieval (IR), and employing machine learning techniques to learn the ranking function is viewed as a promising approach to IR. In information retrieval, the users'queries often vary a lot from one to another. However most of existing approaches for ranking do not explicitly take into consideration the fact that queries vary significantly along several dimensions. In this paper, query difference is incorporated into ranking by applying query-dependent loss function to the original loss function of RankBoost algorithm. The effectiveness of the system will be tested on LETOR, publicly available benchmark dataset.

1. Introduction

Many applications have ranking as the central issue, such as information retrieval, collaborative filtering, expert finding, data mining, and anti web spam. Recently, "learning to rank" has been one of the most popular research topics in the areas of information retrieval and search engines. When applied to document retrieval, the task of learning to rank is to construct a ranking function for a search engine.

An effective ranking framework is the core component of any information retrieval system and several ranking functions emerged including the Boolean model, the vector spacemodel [8] and BM25. They have the advantage of being fast and produce reasonably good results. When more features become available, however, incorporating them into these models is usually difficult since it requires a significant change in the underlying model. Recently machine learning techniques have also been applied to ranking model construction and supervised learning to rank algorithms can help overcome that limitation. Several methods for learning to rank have been developed. Typical methods include RankSVM[11], RankBoost[6][7], RankNet[1], and some improved methods such as MHR[5], AdaRank[11], and ListNet[5].

Queries vary largely in multiple facets, for example, queries can be navigational, informational, or transactional and the diverse feature impacts on ranking relevance with respect to difference queries.

In this paper different loss functions for different query categories are combined in learning the ranking function. Instead of extracting individual objective for each query, query categorization is used to represent query difference such that each query category stands for one kind of ranking objective.

The rest of this paper is organized as follows. Related work is presented in Section 2. Section 3 presents the system architecture. In section 4 and section 5 , the dataset and evaluation methods that will be used to determine the performance of the system are described. Finally, section 5 presents conclusion of the paper.

2. Related Work

SomnathBanejeeet.al[4] proposed a local learning algorithm based on new similarity measure between queries. Firstly, they defined the principal components for each query. After that, they used an offline method to cluster queries base on their proposed similarity measure

and train a model for each cluster. When a test query is entered, they used the model from the most similar cluster.

Weijian Ni et.al[14] developed a query dependent ranking approach. In their approach, the ranking model of each query consists of a generalizable model and a specific model. During the learning stage, the generalizable and specific models are learned through using structural risk minimization (SRM) inductive principle. At the inference stage, for each new query, several of the most favorable specific models learned from training queries are used to generate its adaptable ranking model.

Lian-Wang Lee et.al[12], also proposed a new framework for query-dependent ranking. They generated individual ranking models from each training queries. When a new query is asked, the retrieved documents of the new query are ranking according to their scores given by a ranking model which is a weighted combination of the models of similar training queries.

XiuboGenget.al[9] developed query-dependent ranking by using K-Nearest Neighbor (KNN) method. They create a ranking model for a given query by using the labeled neighbors of the query in the query feature space and then rank the documents with respect to the query using that model.

In most of the previous works, the ranking features of top-T documents retrieved by a reference model are used to generate query features for each query. Since the reference model is one of the ranking features, the generated query features may biased towards the reference model. This system tend to consider all ranking features equally by computing the average score of all ranking features and used the ranking features of documents which have the highest average score to generate query feature.

Since most of the previous query-dependent ranking approaches need to be trained many model for each query category, they training time of these approaches could be quite large. During testing, these approaches need to spend additional time to locate the appropriate model for the test query.

So that the proposed system tries to minimize the loss function by categorizing the queries in learning, it needs to be trained only one model for all query categories. Therefore, the training time and testing time can be reduced.

3. System Architecture

The system consists of three main components : Query feature generation module, Ranking model construction module and testingmodule.Figure.1 shows the architecture of the proposed system.

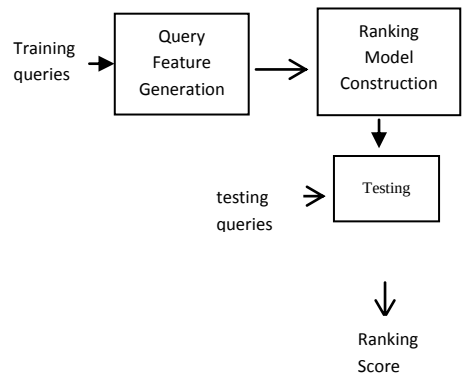

Figure 1. Architecture of the proposed system

3.1. Query feature generation

In order to achieve high accuracy, query features used in the method are important. A query is associated with a list of retrieved documents, each of which can be taken as an observation about the query. In this system, ranking features of the retrieved documents are used for each query. To extract the query features for each query, the system use the top T documents which have the highest average scores of its documents features. The system use the average scores of all features so that the system can consider all ranking features equally without biasing to only one feature. For each query $q \in Q$ in the training data, the system find top T documents $D_q = \{d^1, d^2, d^3 \ldots d^T\}$ for q. We use the ranking features of these documents to extract the features of the query. Each document $d^i \in D_q$ is represented with n ranking features $d^i_x = \{ d^i_{x1}, d^i_{x2} \ldots, d^i_{xn}\}$. To represent q in a feature space we

aggregate the ranking features of all documents in D_q. We use variance of ranking features values as follow:

For each q∈Q,

$$\boldsymbol{\mu_{x_j}(q)} = \frac{\sum_{i=1}^{T} d_{x_j}^{i}}{T} \quad (1)$$

where$^{i}_{xj}$means the jth feature value of ith document for query q.

For each q∈Q,

$$\boldsymbol{\sigma_{x_j}^2} = \sum_{i=1}^{T} \frac{\left(d_{x_j}^{i} - \mu_{x_j}(q)\right)^2}{T} \quad (2)$$

So we can define the query q in terms of aggregated variance values of each feature over q's top T retrieved documents.

3.2. Ranking Model Construction

The system considerquery categorization as hidden information and optimizes the ranking function jointly with query categorization. In terms of search intentions, queries are usually classified into three major categories according to Broder'staxonomy[2] : navigational, informational and transactional. So the system classify queries into three categories, i.e., C={ C_I, C_T, C_N) ,where C_I denotes informational queries , C_T denotes transactional queries and C_N denotes navigational queries. Before learning the ranking function to construct the ranking model, query-dependent loss function is defined and learn the ranking function by minimizing the loss function. We modified the method used in [3] in order to learn the ranking function jointly with query categorization as shown in figure (2). Gradient descent method is used in each iteration.

3.2.1. RankBoost

RankBoost[15] is an extension of the boosting philosophy for ranking and operates on document pairs. Suppose $d_i \gg_q d_j$ means that document d_i should be ranked higher than d_j for query q. Consider the model f, where $f(\varphi(q, d_i))$>f(φ, d_j)) means that the model asserts $d_i \gg_q d_j$. Then the loss for a document pair in RankBoost is defined as

$$L(d_i \gg_q d_j) = e^{-(f(\varphi(q,d_i)) - f(\varphi,d_j)))} \quad (3)$$

Consequently, the loss function of RankBoost is the sum of losses for all document pairs:

$$L = \sum_q \sum_{d_i \gg_q d_j} L\,(d_i \gg_q d_j) \quad (4)$$

The loss function of RankBoost is modified by applying query dependent loss function as follows:

$$L(f;q) = \alpha_q \left(\sum_{d_i \gg_{q_I} d_j} e^{-(f(\varphi(q_I,d_i)) - f(q_i,d_j)))}\right) +$$
$$\beta_q \left(\sum_{d_i \gg_{q_T} d_j} e^{-(f(\varphi(q_T,d_i)) - f(q_T,d_j)))}\right) +$$
$$\theta_q \left(\sum_{d_i \gg_{q_N} d_j} e^{-(f(\varphi(q_N,d_i)) - f(q_N,d_j)))}\right) \quad (5)$$

After this, the query-dependent ranking model is learned by minimizing the loss function in eq(5).

3.2.2. Gradient Descent Method

The algorithm is initialized with a guess x_1, a maximum iteration count N_{max} , a gradient norm tolerance $\in_g$ that is used to determine whether the algorithm has arrived at a critical point, and a step tolerance $\in_x$ to determine whether significant progress is being made. It proceeds as follows [10].

1. For t=1,2,…,N_{max}
2. $x_{t+1} \leftarrow x_t - \alpha_t \nabla f(x_t)$
3. If $\|\nabla f(x_{t+1})\| < \in_g$ then return "Converged on critical point"
4. If $\|x_t - x_{t+1}\| < \in_x$ then return "Converged on an x value"
5. If $x_{t+1} > f(x_t)$ then return "Diverging"
6. Return "Maximum number of iterations reached"

The variable α_t is known as the step size and should be chosen to maintain a balance between convergence speed and avoiding divergence. Note that α_t may depend on the step t.

Input : a set of training examples for learning to rank; queries defined by query features
output: parameter vector of the ranking function w, and parameter vector of the query categorization, γ, which minimize the loss function L_f.
Algorithm:
Start with an initial guess, e.g. random values, for w and γ;
Begin
While ($L_f(w_k, r_k)$ - ($L_f(w_{k+1}, r_{k+1})$ >$\in$ do
Step 1 : Learning w
Using gradient descent to minimize L_f with respect to ranking function parameters w, holding the query categorization parameters fixed, i.e.,

$$w_{k+1} \leftarrow arg \min_{w} \sum_{q \in Q} \alpha_{\gamma_k}(q) L(f_{w_k}; q, C_I) + \beta_{\gamma_k}(q) L(f_{w_k}; q, C_T) + \theta_{\gamma_k}(q) L(f_{w_k}; q, C_N))$$

Step 2 : Learning γ Using gradient descent to minimize L_f with respect to query categorization parameters γ, holding the ranking function parameters fixed, i.e;

$$\gamma_{k+1} \leftarrow \arg\min_{\gamma} \sum_{q \in Q} \alpha_{\gamma_k}(q) L(f_{w_{k+1}}; q, C_I) + \beta_{\gamma_k}(q) L(f_{w_{k+1}}; q, C_T) + \theta_{\gamma_k}(q) L(f_{w_{k+1}}; q, C_N))$$

End

Figure 2. Unified Learning Method

3.3. Testing

In testing module the ranking model produced by ranking model construction module is used to produces the ranking results for the test query.

4. Dataset

LETOR is a benchmark dataset for research on ranking [13]. The performance of thesystem will be evaluated on TREC 2003, TREC 2004 and OHSUMED, which are included in LETOR 3.0 dataset.According to [13] the statistics of the datasets from the LETOR 3.0 is described in table 1.

Table 1. Statistics of the datasets from LETOR

	Queries	Rel: levels	features
TREC 2003	350	2	64
TREC 2004	225	2	64
OHSUMED	106	3	45

5. Performance Metric

The performance of the system which employed query-dependent loss function will be evaluated by using three IR evaluation measures**:** Precision, Mean Average Precision(MAP) and Cumulative Gain (NDCG) which are widely used in information retrieval.

(i). Precision

For a given query, its precision of the top n results of the ranking lists is defined as:

$$P@n = \frac{\#\textbf{relevant results in top n results}}{\boldsymbol{n}} \quad (6)$$

(ii). Mean Average Precision(MAP)

Given a query, its average precision can be computed as follows:

$$AP(q) = \frac{\sum_{i=1}^{n} P@n * rel(n)}{\#totalrelevantresultsforthequery} \quad (7)$$

where N is the number of retrieved documents and rel (n) is either 1 or 0, indicating that n^{th} document is relevant or not to the query. MAP for a set of queries is the mean of the average precision scores for each query.

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \quad (8)$$

where Q is the number of queries.

(iii) . Normalized Discounted Cumulative Gain (NDCG)

For a query, the NDCG of its ranking list at position n is calculated as follows:

$$NDCG(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{log(1+j)} (9)$$

wherer (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.

The performance of the system will be evaluated by comparing with pervious query-dependent ranking approaches based on these three evaluation measures.

5. Conclusion

The system developed query-dependent ranking model based on query-dependent loss function by conducting learning of the ranking model jointly with that of query categorization. The system can achieve better ranking performance because it combine the loss function for all query categories together on each query, but they are weighted according to the query's soft categorization. In addition, the training and testing time can be shorter compared to other query-dependent ranking approach because the system need to train only one model for all query categories while other query dependent ranking approach need to be trained many models.The system is now in implementation state and the training and testing time of the system will be compared with other query-dependent ranking approach.

References

[1]B. Chris, S. Tal, R. Erin, L. Ari, D. Matt, H. Nicole, H. Greg , "Learning to rank using Gradient Descent", Proceeding of the 22nd International Conference on Machine learning, Bonn, Germany, 2005.

[2] B.Andrei"A taxonomy of web search"

[3]B.Jiang, L.Tie-Yan, Z.Hongyuan "Ranking with Query-Dependent Loss for Web Search"

[4]B.Somnath, D.Avinava , M. Jinesh, C. Soumen, "Efficient and accurate local learning for ranking", copyright 2009 ACM.

[5]C. Zhe, Q. Tao, L.Tie-Yan, T. Ming-Feng, L.Hang, "Learning to Rank: From Pairwise Approach to Listwise Approach" , Proceedings of 24th International Conference on Machine Learning, Corvallis, 2007.

[6]D. Kevin , K. Katrin, "Learning to rank with partially-labeled data", SIGIR'08, Singapore , July20–24,2008.

[7]F. Yoav, I. Raj, E. S. Robert, "an efficient boosting algorithm for combining preferences", Journal of machine learning research 4 (2003) 933-969.

[8] G.Salton, M.J.McGill, "Introduction to Modern Information Retrieval"

[9]G. Xiubo, L. Tie-Yan, Q .Tao, "Query Dependent Ranking Using K-Nearest Neighbor", SIGR'08, Singapore, July 20-24, 2008

[10]H.Kris, Gradient Descent

[11]Jun Xu, Hang Li, "AdaRank: A Boosting Algorithm for Information Retrieval", SIGR'07, July 23-27, 2007 ,The Netherlands.

[12]L. Lian-Wang, J. Jung-Yi, W. ChunDer, L. Shie-Jue, "A Query-Dependent Ranking approach for search engines",2009 Second international workshop on Computer Science and Engineering.

[13] L.Tie-Yan, X.Jun, Q.Tao, X.Wening, L.Hang , "LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval".

[14]N. Weijian, H. Yalou, X. Maoqiang, "A Query Dependent Approach to Learning to Rank for Information Retrieval" , The Ninth International Conference on Web-Age Information Management, copyright 2008 IEEE.

[15] Q.Tao, L.Tie-Yan, T.Ming-Feng, Z.Xu-Dong, L.Hang "Learning to Search Web Pages with Query-Level Loss Functions"