

Query Dependent Ranking for Information Retrieval Based on Query Clustering

Pwint Hay Mar Lwin

University of Computer Studies, Yangon, Myanmar

Article Info

Article history:

Received Sep 30th, 2012

Revised Nov 15th, 2012

Accepted Nov 25th, 2012

Keyword:

Information Retrieval

LETOR

Query clustering

Ranking function

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. In this paper we take into account the diversity of query type by clustering the queries. Instead of deriving a single function, this system attempt to develop several ranking functions based on the resulting query clusters in the sense that different queries of the same cluster should have similar characteristics in terms of ranking. Before the queries are clustered, query features are generated based on the average scores of its associated retrieved documents. So, for each query cluster, there will be its associated ranking model. To rank the documents for a new query, the system first find the most suitable cluster for that query and produce the scoring results depend on that cluster. The effectiveness of the system will be tested on LETOR, publicly available benchmark dataset.

Copyright © 2013 Institute of Advanced Engineering and Science.

All rights reserved.

Corresponding Author:

Third Author,

Pwint Hay Mar Lwin

University of Computer Studies, Yangon, Myanmar

Email: pwinthaymarlwin.phml@gmail.com

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 space model[7] 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[9], RankBoost[5][6], RankNet[1], and some improved methods such as MHR[4], AdaRank10], and ListNet[4].

Using a single ranking model may not be appropriate, particularly for web search, since queries vary largely in multiple facets, for example, queries can be navigational, informational, or transactional. These various types of query difference make it difficult to build a single general ranking function for all kinds of queries. This is because the diverse feature impacts on ranking relevance with respect to difference queries. In this paper we will consider the query diversity in ranking by clustering the queries and derive separate model for each cluster.

The rest of this paper is organized as follows. Related work is presented in Section 2. Section 3 presents the system framework. In section 4, 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

Jiang Bian et.al[2] proposed the divide and conquer approach for ranking specialization. They divided the problem of learning one single ranking model for all training queries into a set of sub-problems for learning multiple models. They also proposed global loss function to learn multiple ranking models simultaneously. For a new query, the ranking result is produced by combining the corresponding ranking results of the models whose corresponding query topic hold the H highest correlation values with that new query. Somnath Banejee et.al[3] 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[13] 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[11], 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. Xiubo Geng et.al[8] 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.

3. Proposed System

Figure 1, shows the architecture of the proposed system.

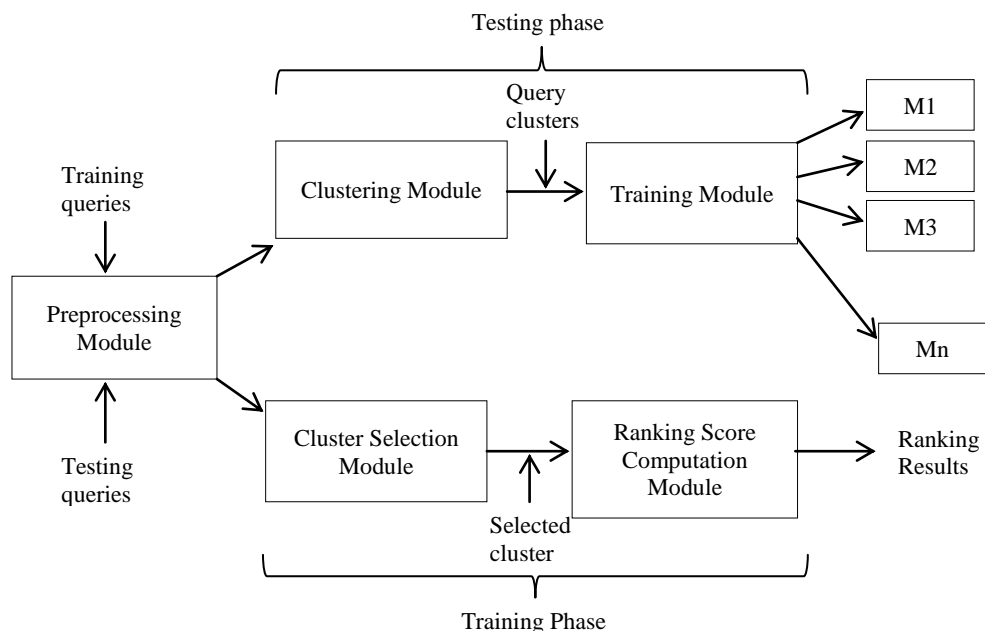


Figure 1. System Architecture

3.1 Preprocessing Phase

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 $D_q = \{d^1, d^2, d^3 \dots d^T\}$ 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 $d_{x^i} = \{d_{x_1^i}, d_{x_2^i}, \dots, d_{x_n^i}\}$. 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:

$$\text{For each } q \in Q, \\ \mu_{x_j}(q) = \frac{\sum_{i=1}^T d_{x_j^i}}{T} \quad (1)$$

where $d_{x_j^i}$ means the j^{th} feature value of i^{th} document for query q.
For each $q \in Q$,

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

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 Training Phase

3.2.1 Clustering Module

After the query features are generated, bisecting k-means clustering algorithm is applied to identify the query clusters.

The bisecting k-means algorithm is a straight forward extension of the basic k-means algorithm that is based on a simple data: to obtain k clusters, split the set of all points into two clusters, select one of these clusters to split, and so on, until k clusters have been produced.

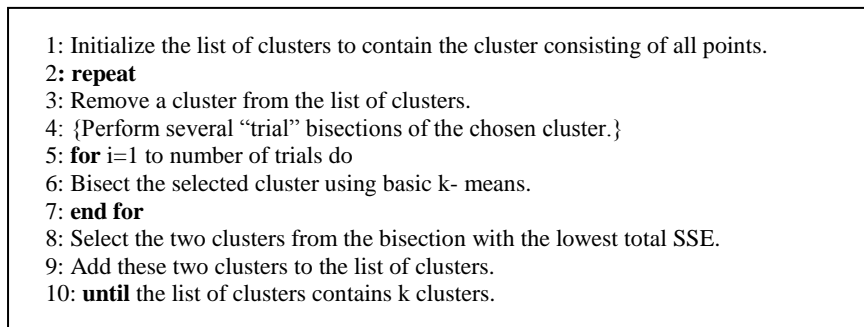


Figure 2. Bisecting k-means Clustering Algorithm

3.2.2 Training Module

The system create separate model for each query cluster. To construct a ranking model for each cluster the system adopts RankSVM [2][9]. Ranking SVM is a generalization of classical SVM formulation that learns over pairwise preferences, rather than binary labeled data. Instead pairwise preferences can implicitly encode the structure of ranking problems, and therefore learning an SVM over such pairwise preferences is typically more effective when used for ranking since its objective function tends to more in line with standard information retrieval metrics, such as precision, mean average precision and F1 score, which is the harmonic mean of precision and recall.

Formally, the ranking SVM is formulated as a quadratic programming problem that has the following form:

$$\min \frac{1}{2} \|w^2\| + C \sum_{i,j} \varepsilon_{i,j} \\ \text{subject to } (w \cdot x_i - w \cdot x_j) \geq 1 - \varepsilon_{i,j} \quad \forall (i,j) \in \rho \\ \varepsilon_{i,j} \geq 0 \quad \forall (i,j) \in \rho \quad (3)$$

Where w is the weight vector being fit, ρ is the set of pair-wise preferences used for training, and C is a tunable parameter that penalizes misclassified input pairs. Once a weight vector w is learned, we can score the documents for unseen queries. These scores can then be used to rank documents.

3.3 Testing Phase

In the testing process, the most suitable cluster for the test query is identified by finding the closest centric c . To find the closest centroid for the test query the system use the Euclidian distance.

$$c^* = \arg \min(\text{dist}(t, x^{-i})) \quad (4)$$

where x^{-i} is the centroid of i th cluster and $i=1,2,\dots,k$.

$$\text{dist}(x^{-i}, t) = \sqrt{(x_1^{-i} - t_1)^2 + (x_2^{-i} - t_2)^2 + \dots + (x_n^{-i} - t_n)^2} \quad (5)$$

After the closest cluster is found, the system retrieves the model with respect to that cluster and produces the ranking results for the test query by using the retrieved model. Figure 3 describe the algorithm of the system.

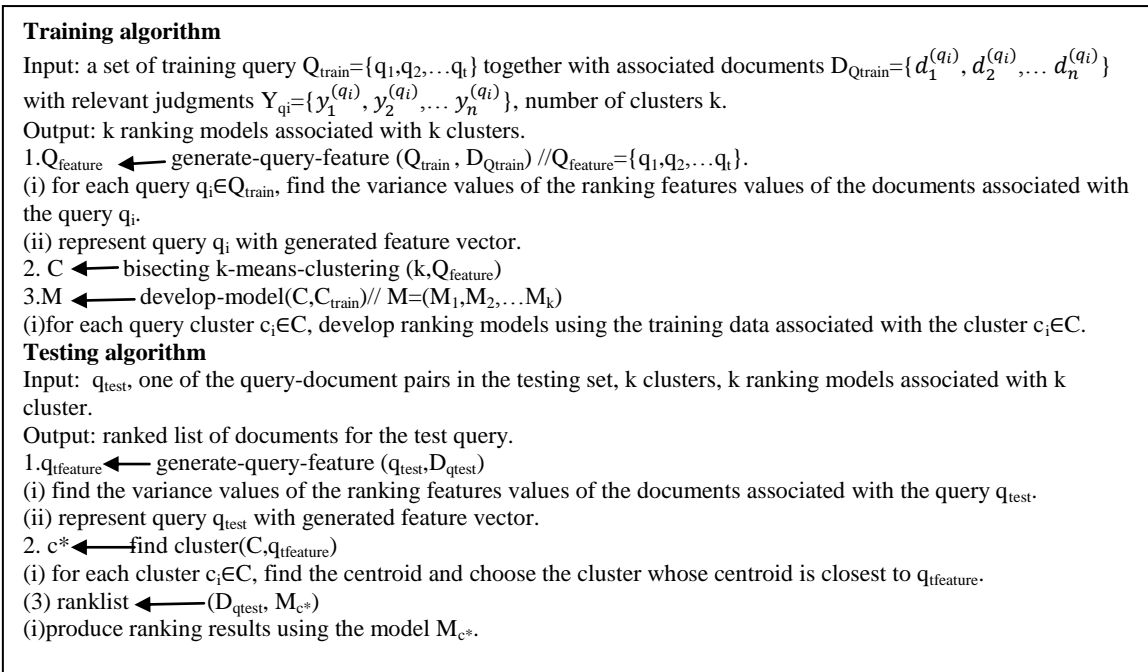


Figure 3. Algorithm of the System

4. DATASET AND PERFORMANCE METRIC

The system will be evaluated on the LETOR benchmark dataset. The evaluation tools provided by LETOR are utilized to evaluate the effectiveness of the proposed system.

4.1. Dataset

The system will be evaluated on TREC 2003, TREC 2004 and OHSUMED, which are included in LETOR 3.0 dataset [12]. The statistics of the datasets from the LETOR 3.0 is described in table 1.

Table 1. Statistics of the Dataset from LETOR

Symbol	Queries	Relevance Levels	Features
TREC 2003	350	2	64
TREC 2004	225	2	64
OHSUMED	106	3	45

Figure 4 shows the sample data from LETOR.

0	qid:1	1:3.00000000	2:2.07944154	3:0.42857143	4:0.40059418	5:37.33056511
2	qid:1	1:0.00000000	2:0.00000000	3:0.00000000	4:0.00000000	5:37.33056511
2	qid:1	1:4.00000000	2:2.77258872	3:0.33333333	4:0.32017083	5:37.33056511
0	qid:1	1:0.00000000	2:0.00000000	3:0.00000000	4:0.00000000	5:37.33056511
1	qid:1	1:1.00000000	2:0.69314718	3:0.14285714	4:0.13353139	5:37.33056511
0	qid:1	1:0.00000000	2:0.00000000	3:0.00000000	4:0.00000000	5:37.33056511
0	qid:1	1:1.00000000	2:0.69314718	3:0.50000000	4:0.40546511	5:37.33056511
0	qid:1	1:3.00000000	2:2.07944154	3:0.60000000	4:0.54696467	5:37.33056511
0	qid:1	1:0.00000000	2:0.00000000	3:0.00000000	4:0.00000000	5:37.33056511
0	qid:1	1:1.00000000	2:0.69314718	3:0.33333333	4:0.28768207	5:37.33056511
0	qid:1	1:0.00000000	2:0.00000000	3:0.00000000	4:0.00000000	5:37.33056511
1	qid:1	1:0.00000000	2:0.00000000	3:0.00000000	4:0.00000000	5:37.33056511
1	qid:1	1:2.00000000	2:1.38629436	3:0.28571429	4:0.26706279	5:37.33056511

Figure 4. Sample Data from LETOR

4.2 Evaluation Measure

The system will be evaluated on TREC 2003, TREC 2004 and OHSUMED, which are included in LETOR 3.0 dataset [12]. The statistics of the datasets from the LETOR 3.0 is described in table 1. The system will be evaluated using three common IR evaluation measures supported by LETOR.

1) Precision

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

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

2) Mean Average Precision (MAP)

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

$$AP(q) = \frac{\sum_{i=1}^n P@n * rel(n)}{\text{\#total relevant results for the query}} \quad (7)$$

where N is the number of retrieved documents and rel (n) is either 1 or 0, indicating that nth 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.

3) Normalized Discounted Cumulative Gain (NDCG)

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

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

where r (j) is the rating of the jth document in the ranking list, and the normalization constant Z_n is chosen so that the perfect list gets a NDCG score of 1.

5. CONCLUSION

This paper uses a framework for query-dependent ranking model. In this system, instead of learning a single model for all training queries, individual model is developed for each query group that consists of the subset of training queries which have the similar features for ranking. Since the diverse feature impacts on ranking relevance with respect to different queries, the result produces by a single ranking function, while indicating good ranking relevance for a certain type of queries may not be able to achieve similar performance for other type of queries. By using separate model for each query cluster we can use separate features and training data for learning the ranking model for each query cluster. Therefore, we can apply useful information of the similar queries and avoiding negative effect of dissimilar ones. So we can achieve better ranking performance than a single model approach for each query cluster without hurting others. On the other hand, it may use only a small part of training data for learning each model; it may cause declining accuracy due to the lack of enough training examples. The evaluation will be done on LETOR bench mark data set.

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. W.-K. Chen, *Linear Networks and Systems* (Book style).Belmont, CA: Wadsworth, 1993, pp. 123–135.
- [2] B.Jiang, L. Xin, L. Fan, Z.Hongyun, "Ranking Specialization for web search: A Divide and Conquer Approach by Using Topical RankSVM" WWW 2010, Raleigh, North Carolina, USA , April 26-30.
- [3] B.Somnath, D.Avinava , M. Jinesh, C.Soumen, "Efficient and accurate local learning for ranking", copyright 2009 ACM.
- [4] 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.
- [5] D. Kevin , K. Katrin, "Learning to rank with partially-label data", SIGIR'08, Sigapore, July 20-24, 2008.C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.
- [6] F. Yoav, I. Raj, E. S. Robert, "an efficient boosting algorithm for combining preferences", Journal of machine learning research 4 (2003) 933-969.
- [7] G.Salton, M.J.McGill, "Introduction to Modern Information Retrieval"
- [8] G. Xiubo, L. Tie-Yan, Q .Tao, "Query Dependent Ranking Using K-Nearest Neighbor", SIGR'08, Singapore, July 20-24, 2008
- [9] H. Ralf, G. Thore, O. Klaus, "Large Margin Rank Boundaries for Ordinal Regression"
- [10] Jun Xu, Hang Li, "AdaRank: A Boosting Algorithm for Information L.Tie-Yan, X.Jun, Q.Tao, X.Wening, L.Hang , "LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval".
- [11] 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.
- [12] L.Tie-Yan, X.Jun, Q.Tao, X.Wening, L.Hang , "LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval".
- [13] 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.
- [14] Q. Tao, L. Tie-Yan, L. Wei, Z. Xu-Dong, W. De-Sheng, L. Hang, "Ranking with multiple hyperplanes" SIGR'07, The Netherlands,July 23-27, 2007.

BIBLIOGRAPHY OF AUTHORS



I received B.C.Sc, B.C.Sc(Hons:) and M.C.Sc degrees in Computer Science from University of Computer Studies, Yangon(UCSY) in 2005,2006 and 2009 respectively. During 2007-2009, I served as a tutor in Computer Application Department of University of Computer Studies,Mawlamyine. From 2009 to 2010, I attened PH.D coursework in UCSY. Now, I'm doing my research which is related with information retrieval.