Query Dependent Ranking based on PCA-based Query Representation

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

Abstract

Ranking is a crucial part of information retrieval. Queries describe the users' search intent and therefore they play an essential role in the context of ranking for information retrieval. The diverse feature impacts on ranking relevance with respect to different queries. This paper tends to consider query difference in learning ranking function by clustering the queries where each query cluster represents a group of queries which have the similar set of important features for measuring ranking relevance. The success of clustering usually depends on the representation of the data. The query features are generated based on the ranking features values of query-document pair and Principal Component Analysis (PCA) is used to construct the representation of query. To cluster the queries, bisecting k-means clustering algorithm is used. RankSVM algorithm is used for model construction.

1. Introduction

Ranking function is a crucial part of any information retrieval system, which orders the retrieved documents according to the decreasing relevance to the query. 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.

Learning to rank represents a category of effective ranking methods for information retrieval (IR) systems. Given training data, in the form of a set of queries each associated with a search results labeled by relevance degree, learning to rank returns a ranking function that can be used to order search results for future queries. Several methods for learning to rank have been developed. Typical methods include RankSVM[12], RankBoost[5][6], RankNet[1], and some improved methods such as MHR[4], AdaRank[9], and ListNet[4].

Queries in IR may vary largely in semantics and the users' intentions they represent, in forms they appear, and in number of relevant documents they have in the document repository. For example, queries can be navigational, informational or transactional. If one can successfully leverage query differences in the process of learning to rank, one may have the opportunities to achieve better ranking performances in both training and test processes [14].

So, this paper considers 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 architecture. 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[10], 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. System Architecture

The system includes preprocessing phase, training phase and testing phase.

Figure.1 shows the architecture of the system.

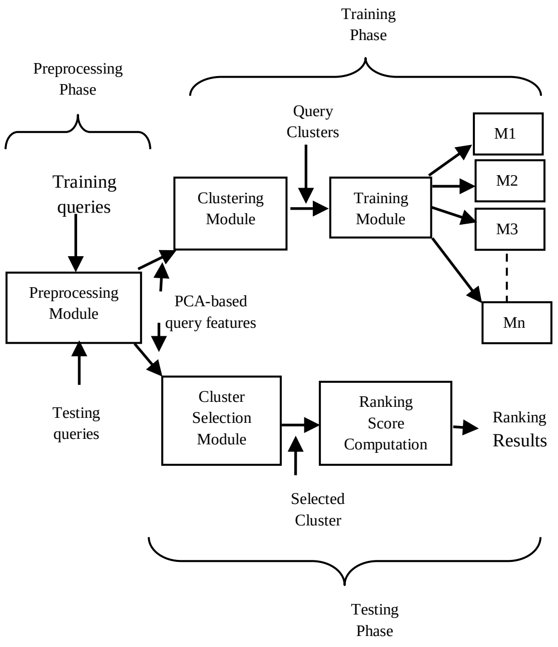
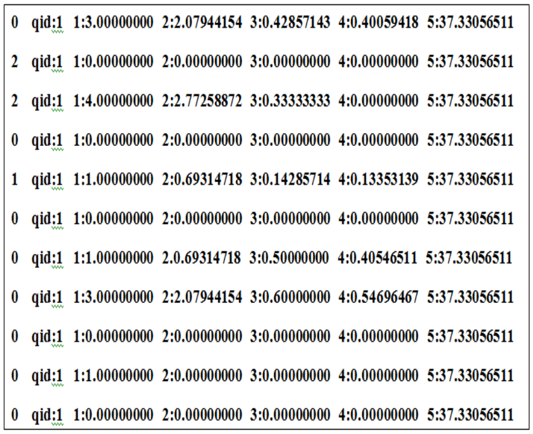


Figure 1. System Architecture

3.1. Preprocessing Phase

A query is associated with a list of retrieved documents, each of which can be taken as an observation about the query. The ranking features of query-document pair of (q,d_i) are defined as a feature vector $x^{qd_i} = (x_1^{qd_i}, x_2^{qd_i}, \ldots\ldots, x_m^{qd_i})$ where m is the number of ranking features.



```
0  qid:1  1:3.0000000  2:2.07944154  3:0.42857143  4:0.40059418  5:37.33056511
2  qid:1  1:0.0000000  2:0.0000000  3:0.0000000  4:0.0000000  5:37.33056511
2  qid:1  1:4.0000000  2:2.77258872  3:0.33333333  4:0.0000000  5:37.33056511
0  qid:1  1:0.0000000  2:0.0000000  3:0.0000000  4:0.0000000  5:37.33056511
1  qid:1  1:1.0000000  2:0.69314718  3:0.14285714  4:0.13353139  5:37.33056511
0  qid:1  1:0.0000000  2:0.0000000  3:0.0000000  4:0.0000000  5:37.33056511
0  qid:1  1:1.0000000  2.0.69314718  3:0.50000000  4:0.40546511  5:37.33056511
0  qid:1  1:3.0000000  2:2.07944154  3:0.60000000  4:0.54696467  5:37.33056511
0  qid:1  1:0.0000000  2:0.0000000  3:0.0000000  4:0.0000000  5:37.33056511
0  qid:1  1:1.0000000  2:0.0000000  3:0.0000000  4:0.0000000  5:37.33056511
0  qid:1  1:0.0000000  2:0.0000000  3:0.0000000  4:0.0000000  5:37.33056511
```

Figure 2. Sample Dataset

Figure 2 shows the sample data of the dataset for query 1 and its associated retrieved documents. The first attribute describes the relevance score of the document. The second attribute shows the query id and the remaining attributes are the ranking features of the document for the query.

According to the figure, each query is represented by its retrieved documents with their associated ranking features. In the original dataset, each query contains about 1000 retrieved documents and so each query is represented with 1000 records. In order to represent the query with only one record, the scores of each ranking feature is sorted and used only top-h highest scores of each feature in data preprocessing. Principal Component Analysis is a standard technique for data preprocessing which reduces the number of dimensions as much variance as possible. The query represent with top-h highest scores is applied to PCA. Figure 3 describes the processing step of PCA.

Step 1 : Get the input data (the query represents with top-h highest scores of each feature dimension)
Step 2 : Subtract the mean value from each of the data dimension
Step 3: Calculate the covariance matrix.
Step 4 : Calculate the eigenvectors and eigenvalues of the covariance matrix
Step 5 : Choosing components and forming a feature vector
Step 6 : Deriving the new data set

Figure 3. Processing Steps of PCA

The mean value of the feature vectors which are produced by PCA is taken as PCA based query feature and now each query can be represented with one record. After this step both the number of dimensions and the number of records can be reduced.

3.1. Training Phase

Training phase consists of clustering module and training module.

3.1.2. 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.

The main experimental question is, will clustering make it easier to locate the best cluster from which to use the trained model, and whether clustering will smear together queries that are not quite similar enough, so as to reduce the efficiency of cluster models for any specific test query.

```
Begin
Clist $\longleftarrow$ cluster containing all points
Repeat
{
    pick a cluster C from Clist
   for i=1 to num-of-trials
{
    bisect  C using basic K-means
}
Clist $\longleftarrow$ two clusters from the bisection which
have
the lowest SSE
}Until the desired number of clusters is reached
```

Figure 4. Bisecting k-means Clustering Algorithm

3.1.3. Training Module

Ranking model for each query is constructed by using RankSVM[12]. Ranking SVM is a generalization of classical SVM formulation that learns over pairwise preferences, rather than binary labeled data. 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:

$$\min \frac{1}{2}\|w^2\| + C\sum_{i,j} \varepsilon_{i,j}$$

$$\text{subject to } (w.x_i - w.x_j) \geq 1 - \varepsilon_{i,j} \quad \forall(i,j) \in \rho$$

$$\varepsilon_{i,j} \geq 0 \qquad \forall(i,j) \in \rho \qquad (1)$$

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.

3.2. Testing Phase

Testing phase consists of clustering selection module and ranking score computing module.

In cluster selection module, 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.

$$c^* = arg\ min\left(dist(t, x^{-i})\right) \qquad (2)$$

where x^{-i} is the centroid of ith cluster and i=1,2,…k.

$$dist(x^{-i}, t) = \sqrt{(x_1^{-i} - t_1)^2 + (x_2^{-i} - t_2)^2 + \cdots + (x_n^{-i} - t_n)^2} \quad (3)$$

After the closest cluster is found, the system retrieves the model with respect to that cluster and computes the ranking scores for the test query by using the retrieved model.

4. Dataset and Performance Metric

The accuracy of the ranking models will be evaluated on the LETOR benchmark dataset[11]. The evaluation tools provided by LETOR are utilized to evaluate the effectiveness of the proposed system.

4.1. Dataset

The experiments will be evaluated on TREC 2003, TREC 2004 and OHSUMED, which are included in LETOR 3.0 dataset [11]. 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

4.2. Evaluation Measures

The experimental results will be conducted using three common IR evaluation measures supported by LETOR.

(i) Precision

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

$$P@n = \frac{\#relevant\,results\,in\,top\,n\,results}{n} \qquad (4)$$

(ii) Mean Average Precision(MAP)

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

$$AP(q) = \frac{\sum_{i=1}^{n} P@n * rel(n)}{\#totalrelevantresultsforthequery} \qquad (5)$$

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.

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \qquad (6)$$

where Q is the number of queries.

(iii) Normalized Discounted Cumulative Gain (NDCG)

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

$$NDCG(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{\log(1+j)} \qquad (7)$$

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 and Discussion

In this paper, 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 feature for ranking. 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. As a result, useful information of the similar queries can be applied and avoiding negative effect of dissimilar ones. So the better ranking performance can be achieved for each query cluster without hurting others. On the other hand, the ranking accuracy of the system depends on how the queries are clustered and the success of a clustering application usually depends critically on the representation of the data. Therefore it is very important that how the queries are represented.

PCA constructs a set of uncorrelated directions that are ordered by their variance. In many cases, the directions with the most variance are the most relevant to the clustering. By applying PCA to represent the query before clustering, it can be possible that the clustering quality can be improved and therefore the overall ranking performance of the system can also be improved.

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. Jiang, L. Xin, L. Fan, Z. Zhaohui, Z. Hongyuan, "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-labeled data", SIGIR'08, Singapore , July20–24,2008.

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

[10] 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.

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

[12] M.Donald, K.Tapas, "Machine Learned Sentence Selection Strategies for Query-Biased Summarization"

[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] T.Yan Liu, "Learning to Rank for Information Retrieval", Springer.