Query-Dependent Ranking for Web Information Retrieval

Pwint Hay Mar Lwin, Nang Saing Moon Kham
pwinthaymarlwin.phml@gmail.com, moonkham.ucsy@gmail.com

Abstract

In information retrieval, the users' queries often vary a lot from one to another. Most IR systems use a single fixed ranking strategy to support the information seeking task of all users for all queries irrespective of the heterogeneity of end users and queries.

The main problem for this work is that no single ranking strategy performs the best for all queries. This work considers query difference in developing ranking function by clustering the query. Then the cluster membership information is combined into the learning process of the ranking function in order to combine the ranking risks of all training examples with different weights according to the training query's similarity to different query cluster for ranking model construction.

To verify the benefit of the proposed query-dependent ranking system experiments were conducted on TREC 2003 and TREC 2004 datasets in LETOR (Learning To Rank) package. The ranking accuracy of the system is evaluated by Normalized discount cumulative gain (NDCG)evaluation metric.

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 [5] 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[13], RankBoost[3][4], RankNet[1], and some improved methods such as MHR[3], AdaRank[9], and ListNet[3].

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 query diversity is considered 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, evaluation methods that will be used to determine the performance of the system and the evaluation results of the proposed system are described. Finally, section 5 presents conclusion of the paper.

2. Related Work

Somnath Banejee et.al[2] 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[12] 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[6] 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. . The idea has been tested with RankingSVM and the experimental results show significant performance improvements as compared to using one single ranking function and using query classification-based query dependent ranking.Used the similar idea to perform query-dependent ranking.

Heli Sun et.al [7] proposed an alternative approach to RankSVM named QoRank, which performs the learning task dependent on queries. Instead of using a single hyperplane to rank instances to rank instances belonging to different ranks and queries which would make compromises among the cases and result in lower ranking accuracy, it trains multiple query-dependent hyperplanes as the basic decision functions and aggregates the rank lists from multiple hyperplanes as the final ranking. On the one hand, QoRank constructs hyperplanes depending on query and thus is able to provide a better separation of the instances of different queries. On the other hand, it separates the training instance into small pieces and it gains the lowest training complexity. They also proposed a LSE (least-squares estimation)-based weighted method for ranking aggregation, and the method assigns different weights to hyperplanes with weights reflecting the ranking accuracy. Compared with traditional ranking model BM25 and other learning to rank methods, QoRank achieved a good balance of accuracy and complexity.

3. Overview of the System

The system consists of four major components: Preprocessing Module, Query Clustering Module, Ranking Model construction Module, and Testing Module.

Preprocessing Module: In a typical training set of learning to rank dataset, each training query is represented by ranking feature vectors of query-document pairs. In order to cluster the query into different query clusters by using the clustering algorithm, it needs to be represented the queries with the appropriate feature vectors for clustering. So, in the preprocessing phase, the top-n representative documents are selected for each query and the ranking features of these documents are aggregated into query feature vector.

Clustering Module: After generating query features, the Fuzzy C-means clustering algorithm is employed to obtain query categorization information for each query. After this step, the membership probability of each query into different clusters is obtained.

Ranking Model Construction Module: In this module the membership information of each query calculated by the clustering module is used to construct the query-dependent ranking model. The query dependent ranking model is constructed by combining the query categorization information into the learning process of AdaRank when learning the ranking function.

Testing Module: The testing module reports the performance of the resulting query dependent ranking model. Figure 1 shows the system overview of the proposed system.

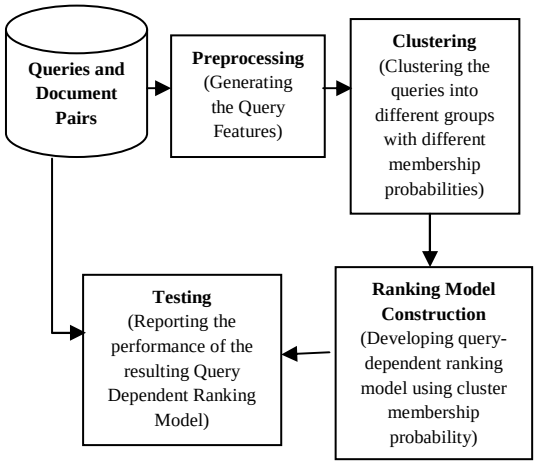

Figure 1. Overall Structure of the Proposed System

3.1 Preprocessing Module

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.

3.1.1 Calculation of Feature Importance Score

Firstly, an important score is assigned to each feature. Specifically, an evaluation measure Normalized Discounted Cumulative Gain (NDCG) is used to compute the importance score. In the former, document instances for each query are ranked using the feature, evaluate the performance in terms of the measure, and then take the evaluation result as the importance score.

After calculating the feature importance score, the most important feature which has the highest impact on accurately ranking the documents associated with each query can be selected.

3.1.2 Generation of Query Features

After selecting the most important feature for each query, the query can be compressed into a query feature vector, where each feature value is obtained by the aggregate of its corresponding features of most influential documents in the query. The process begins with sorting of documents associated with the query according to relevance score of the most important feature of that query.

The system ranks the documents associated with each query by using that feature and aggregate ranking features of top-10 documents (most influential document instances). For each query $q \in Q$ in the training data, the system finds top 10 most influential documents $D_q = \{d^1, d^2, d^3 \dots D^{10}\}$ for q. 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 the feature space, the mean of ranking features values of most influential document instances are taken as follows:

For each $q \in Q$,

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

where $d_{x_j}^i$ means the j^{th} feature value of i^{th} influential document for query q.

So, the query q can be defined in terms of aggregated mean values of each feature over $q's$ top 10 most influential documents.

For the queries which have more than one important feature, the average ranking scores of these features are firstly calculated and the top-10 ranked documents are used to generate the query feature vector.

Figure 2 shows the flowchart of the query feature generation process.

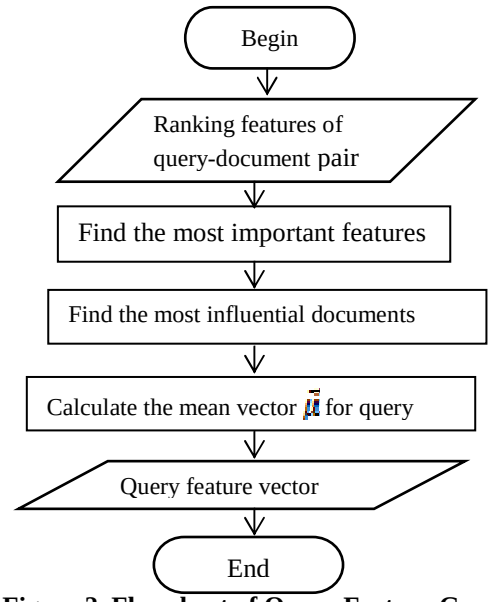

Figure 2. Flowchart of Query Feature Generation Process

3.2 Query Clustering

After generating query features, fuzzy c-means clustering algorithm is employed to obtain query clustering information. In non-fuzzy or hard clustering, data is divided into crisp clusters, where each data point belongs to exactly one cluster. In fuzzy clustering, the data points can belong to more than one cluster, and associated with each of the points are membership grades which indicate the degree to which the data points belong to the different clusters.

3.2.1 Fuzzy C-means Clustering

Fuzzy C-Means [8] (also called Fuzzy K-Means) is an extension K-Means, the popular simple

clustering technique. While K-Means discovers hard clusters (a point belong to only one cluster), Fuzzy C-Means is a more statistically formalized method and discovers soft clusters where a particular point can belong to more than one cluster with certain probability. It is an iterative clustering method produces an optimal c partition by minimizing the weighted within group sum of squared error objective function J_{FCM}.

$$J_{FCM} = \sum_{i=1}^{n}\sum_{j=1}^{c}(\mu_{ij})^m d^2(x_i, c_j) \quad (2)$$

Where m is any real number greater than 1, μ_{ij} is the degree of membership of x_i in j[th] cluster, x_i is the i[th] of dimensional measured data. c_j is the d-dimension center of the cluster, and $d^2(x_i, c_j)$ is a distance measure between x_i and cluster centre c_j.The fuzzifier m determines the level of cluster fuzziness. In the limit m=1, the membership μ_{ij} converge to 0 or 1, which implies a crisp partitioning. In the absence of experimentation or domain knowledge, m is commonly set to 2.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update membership μ_{ij} and the cluster center c_j by:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{d(x_i - c_j)}{d(x_i - c_k)}\right)^{\frac{2}{m-1}}} \quad (3)$$

$$c_j = \frac{\sum_1^N \mu_{ij}^m \cdot x_i}{\sum_{i=1}^N \mu_{ij}^m} \quad (4)$$

The iteration will stop when $max_{ij}\left\{\left\|\mu_{ij}^{(k+1)} - \mu_{ij}^k\right\|\right\} < \varepsilon$, where ε is a termination criterion between 0 and 1, whereas k are the iteration steps. This procedure converges to a local minimum or a saddle point of J_{FCM}.

After learning the clustering model, the distribution of queries over different query clusters can be obtained.

3.3 Ranking Model Construction

To construct the ranking model, the cluster information of each query is combined into the loss function of AdaRank when learning the ranking function. AdaRank is an algorithm which can effectively optimize a loss function based on IR performance measures. In fact, it is a kind of listwise learning to rank algorithm that takes a greedy approach to feature selection, by iteratively selecting the feature that most improves retrieval performance in combination with the previously selected features. Features that are not beneficial to the retrieval performance on the training set will not be selected and it calculates feature weights based on their boosted performance on the training queries.

3.3.1 AdaRank Algorithm

AdaRank [9] takes a training set $S = \{(q_i, d_i, y_i)\}_{i=1}^m$ as input and takes the performance measure function *E* and the number of iterations *T* as parameters. AdaRank runs *T* rounds and at each round it creates a weak ranker $h_t(t = 1, \cdots, T)$.Finally, it outputs a ranking model f by linearly combining the weak rankers.

At each round, AdaRank maintains a distribution of weights over the queries in the training data.The distribution of weights at round t is denoted as P_t and the weight on the i^{th} training query q_i at round t as $P_t(i)$. Initially, AdaRank sets equal weights to the queries. At each round, it increases the weights of those queries that are not ranked well by f_t , the model created so far. As a result, the learning at the next round will be focused on the creation of a weak ranker that can work on the ranking of those 'hard' queries.

At each round, a weak ranker h_t is constructed based on training data with weight distribution P_t . The goodness of a weak ranker is measured by the performance measure *E* weighted by P_t:

$$\sum_{i=1}^m P_t(i) E(\pi(q_i, d_i, h_t), y_i) \quad (5)$$

Once a weak learner h_t is built, AdaRank chooses a weight $\alpha_t > 0$ for the weak ranker. Intuitively, α_t measures the importance of h_t. A ranking model f_t is created at each round by linearly combining the weak rankers constructed so far

$h_1 \cdots h_t$ with weights $\alpha_1 \cdots \alpha_t$. f_t is then used for updating the distribution P_{t+1}.

3.3.2 Combination of Query Information into Learning Process of AdaRank

The system takes a training set S performance measure function *E* and the number of iterations *T* as parameters like AdaRank. When creating the weak ranker at each round, instead of setting distribution weights P_t equally to all queries, the system use the cluster membership information of each query corresponding to each query cluster as the weight of the query and learn the weak ranker for cluster k at round t denoted as h_{kt} Then the weak ranker which can give the best performance for all query clusters is chosen as the final weak ranker h_t.

Once a weak learner h_t is built, the system chooses a weight $\alpha_{kt} > 0$ of the weak ranker for cluster k. The final weight of the weak ranker α_t is obtained by averaging the weights of the clusters.

A ranking model f_t is created at each round by linearly combining the weak rankers constructed so far $h_1 \cdots h_t$ with weights $\alpha_1 \cdots \alpha_t$. f_t is then used for updating the distribution P_{t+1}.

3.4 Testing Module

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 [11]. The performance of the system will be evaluated on TREC 2003, TREC 2004 and OHSUMED, which are included in LETOR 3.0 dataset. The descriptions and the numbers of queries and in these three tasks are shown in Table 1 and 2.

Table 1. Description of Three Search Tasks in TREC 2003 and TREC 2004 Web Track

Query task	Description
Topic distillation	Find a list of entry points for good websites that can guide user to page collection which covers the topic
Homepage finding	Find the home page of a given entity such as an person or organization.
Name page finding	Find the page which directly contains the short description of query but not entry point to website

Table 2. Number of Queries in TREC2003 TREC2004 Web Track

Task	TREC 2003	TREC 2004
Topic distillation	50	75
Homepage finding	150	75
Name page finding	150	75

5. Performance Metric and Evaluation

The performance of the system which employed query-dependent loss function will be evaluated by using Cumulative Gain (NDCG) which is widely used in information retrieval.

5.1 Normalized Discounted Cumulative Gain (NDCG)

Normalized Discount Cumulative Gain (NDCG) is an evaluation metric of ranked retrieval results that can handle multiple levels of relevance judgments.

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

$$NDCG(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{\log(1+j)} \quad (6)$$

where r (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.

5.2 Effects of Different Parameter (number of clusters) Settings

In this experiment the effects of different settings of the parameter k, i.e. the number of clusters, on ranking performance by conducting comparison study with varying value of k. Ranking Performance in terms of NDCG and MAP of proposed method on

TREC 2003 and TREC 2004 datasets against varying number of clusters k are shown in Figure 3.

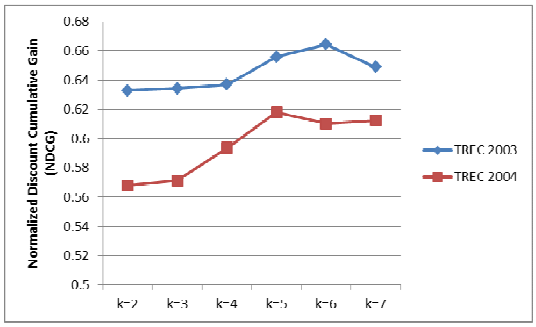

Figure 3 Ranking Performance on TREC 2003 and TREC 2004 Datasets by Varying k Values

5.3 Comparison of Ranking Performance of the Proposed Query Dependent Ranking Model with Query Specific Ranking Model

In this method, the predefined query categorization is applied in order to develop specialized query-dependent ranking model. In LETOR 3.0 dataset, each query can only belong to one category. Thus, this method equals to separating the training queries based on query categorization and training a specialized model for each category of queries. At testing time, the ranking model according to the category of the testing queries is used.

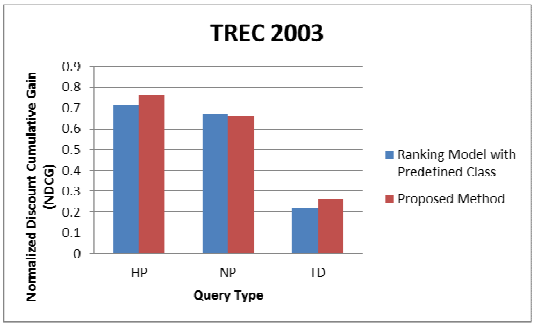

Figure 4. Comparison of Ranking Performance of the Proposed System with Query Specific Ranking Model on TREC 2003

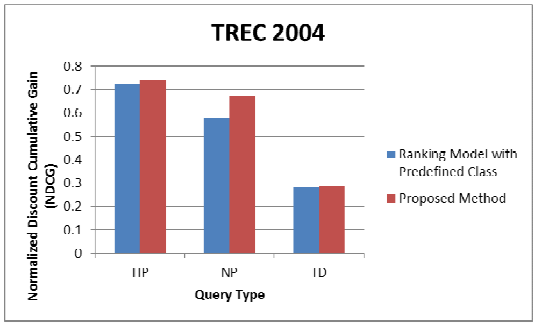

Figure 5. Comparison of Ranking Performance of the Proposed System with Query Specific Ranking Model on TREC 2004

Figure 4 and 5 show the ranking performance of the proposed system compared with query specific ranking approach based of predefined class. According to the figures it can be observed that the system can achieve better ranking results except that the accuracy of the proposed system is slightly lower than that of the ranking model constructed for Name Page Finding Query Type in TREC 2003.

6. Conclusion

This paper has proposed the query-dependent ranking method based on AdaRank by cooperating soft query clustering for solving the ranking problem related to query diversity. The proposed method represents each query by aggregating the ranking features of documents under such query and conduct un-supervised clustering method to identify the query clusters. Moreover, instead of employing hard query clustering and build multiple ranking model corresponding to each query cluster the proposed method can benefit ranking performance by taking advantage of soft query categorization into building the query-dependent ranking model because many queries can fit into more than one clusters. According to the experimental results the proposed query-dependent ranking approach can achieve better ranking relevance than query specific ranking model in terms of NDCG.

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

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

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

[5] G.Salton, M.J.McGill, “Introduction to Modern Information Retrieval”
[6]G. Xiubo, L. Tie-Yan, Q .Tao, “Query Dependent Ranking Using K-Nearest Neighbor”, SIGR’08, Singapore, July 20-24, 2008
[7] H.Sun,J.Huang,B.Feng,“QORank: A Query –Dependent Ranking Model using LSE-based Weighted Multiple Hyperplanes Aggregation for Information Retrieval”, International Journal of Intelligent Systems, Vol.26, pp 73-97,
[8] J.C.Bezdek,R.Ehrlich,W.Full, "FCM: The Fuzzy c-Means Clustering Algorithm" , Computers and Geosciences Vol.10, No.2-3, pp. 191-203, 1984.
[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]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.
[13]R.Herbrich,T.Graepel,K.Obermayer, “Large Margin Rank Boundaries for Ordinal Regression”, Advances in Large Margin Classifiers, pp.115-132 (2000).