Collaborative Ranking: Improving the Relevance for Tail Queries

Ke Zhou
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
kzhou@gatech.edu

Xin Li
Microsoft
1065 La Avenida Street
Mountain View, CA 94043
xinli@microsoft.com

Hongyuan Zha
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
zha@cc.gatech.edu

ABSTRACT

It is well known that tail queries contribute to a substantial fraction of distinct queries submitted to search engines and thus become a major battle field for search engines. Unfortunately, compared with popular queries, it is much more difficult to obtain good search results for tail queries due to the lack of important relevance signals, such as user clicks, phrase matches and so on. In this paper, we propose to utilize the similarities between different queries to overcome the data sparsity problem for tail queries. Specifically, we propose to *jointly* learn query similarities and the ranking function from data so that the relevance signals of different but related queries can be collaboratively pooled to enhance the ranking of tail queries. We emphasize that the joint optimization is critical so that the learned query similarity function can adapt to the problem of learning ranking functions. Our proposed method is evaluated on two data sets and the results show that our method improves the relevance of tail queries over several baseline alternatives.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Retrieval functions*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Collaborative Ranking, Learning to Rank, Relevance, Tail Query, Gradient Boosting

1. INTRODUCTION

It is well known that the query frequency follows a power law distribution — most distinct queries are only submitted for a very few times. These queries are usually called *tail queries* since they correspond to the tail of the power law distribution. At the other end of the distribution are those head queries which are very popular and submitted to search engines by many users.

Although tail queries only contribute to a relatively small fraction of the total traffic to search engines, the relevance of these queries, to a large extent, determines the user experiences. This is because almost all of today's commercial search engines can generally respond to head queries with excellent results. On the other hand, even though tail queries tend to represent the detailed and specific information needs for a particular user, the performance of many search engines is still lacking in serving good results for tail queries, and it has been argued by some that this is one of the reasons that users switch search engines [13]. Consequently, it is the capability of providing good results for tail queries that distinguishes different search engines and in this specific sense, the relevance of tail queries is in fact the battle field of search engines today, making it extremely important to improve the performance for tail queries.

Unfortunately, improving the relevance of tail queries are much more difficult than head queries. We argue that the difficulty of tail queries is largely caused by the inherent data sparsity problem. In this paper, we analyze the issues that limit the relevance of tail queries and address the problem of improving the relevance of tail queries. In particular, we propose a method, Collaborative Ranking, to overcome the data sparsity problem of tail queries. Specifically, we utilize the data from multiple queries in a collaborative manner. This idea allows the problem to be formulated in a principled way as a *joint* supervised learning problem, where the similarity measure between queries and the ranking function to determine the relevance between queries and documents are constructed simultaneously. The evaluations conducted on public domain data sets as well as a tail query data set from a commercial search engine show that the proposed method significantly improves the relevance for tail queries.

2. RELATED WORK

Tail Queries. Several studies have been focused on applications related to tail queries [3,8,10,11]. Moreover, evaluation results of search engines suggest that tail queries contribute to the major differences among search engines, while the results of head queries tend to be the same [13]. These studies show that it is vital and useful to improve the relevance of tail queries.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

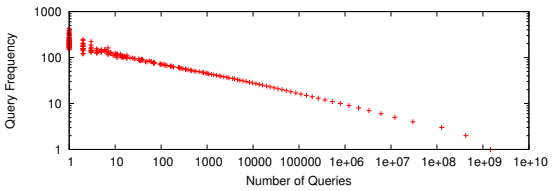


Figure 1: Query frequency with respect to the number of distinct queries

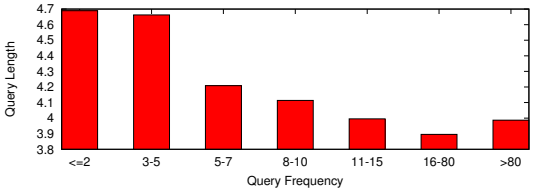


Figure 2: Query length with respect to different levels of query frequency

Query Similarity. It is an important research direction to obtain similarity estimations for queries that accurately reflect the degree of similarities in information needs of users. There are generally two different approaches to obtain query similarity: term based similarities [5,7,12] and user behavior based similarities [1,2,11]. The recent work [9] addresses the problem of learning the linear similarity function of queries in the context of query reformulation. In this work, our goal is not to modify the original queries. On the other hand, we propose to learn the similarity function to overcome the sparsity problem for tail queries based on the gradient boosting framework.

Learning to Rank. The ranking problem is frequently formulated as a supervised machine learning problem [4, 6, 14]. However, existing methods on learning to rank largely ignore the specific issues of tail queries and thus can not provide accurate results for tail queries when the data are sparse.

3. ANALYSIS OF TAIL QUERIES

We investigate a large scale sample of query log from a commercial Web search engine and calculate various statistics of the queries. In Figure 1, we plot the number of queries with different levels of frequency. It can be observed that a large fraction of queries occurs only a few times. For example, nearly 70% of the query occurs only once to the search engine. Another characteristic of tail queries is that they usually carry more specific and detailed information needs from users. In Figure 2, we plot the average number of terms in queries with respect to different frequency levels. It is clear that the lengths of tail queries are generally longer than those of head queries and carry more specific information needs from users.

As a result, the quality of tail queries largely impact the users experiences and choices of search engines. Tail queries represent a large amount of diverse and detailed information needs that are very specific to users. Thus,the results of these queries distinguish search engines from each other.

Unfortunately, the relevance of tail queries is relatively worse than head queries [3]. Specifically, tail queries, by definition, are the queries submitted to search engines very infrequently. Therefore, several problems directly associated with the data sparsity may impact the relevance of these tail queries:

Lacking User Behavior Data. Almost all successful web search engines nowadays rely heavily on user behavior data such as user click-through data. However, they become less useful for tail queries since it is difficult to accurately infer the relevance of a document without collecting sufficient many user behavior data. Thus, these strong signals for head queries are weakened or even missing for tail queries, which makes tail queries very difficult for ranking.

Term Mismatch. Clearly, if a document matches the query exactly, it is likely that the document is relevant to the query. However, it is common that the same information need can be expressed in multiple ways. Therefore, if the information needs are expressed in less frequently used ways, it can become very difficult to find exact matches for the query, which making it difficult the predict the relevances of documents to the query.

In summary, the relevance for tail queries is relatively worse compared with that for head queries. This is a quite unsatisfactory situation considering the importance of tail queries to user experiences of web search engines.

4. COLLABORATIVE RANKING

4.1 Problem Formulation

In this section, we describe CollRank, the proposed collaborative ranking framework, in detail. The basic idea is to make use of similar queries to improve the ranking of tail queries. When computing the relevance score $f(q,d)$ between a tail query q and a document d, we take a few other queries q' into account rather than consider q and d alone. Specifically, we express the ranking function $f(q,d)$ as follows:

$$f(q,d) = h(q,d) + \frac{1}{|S_q|}\sum_{q' \in S_q} s(q,q')h(q',d),$$

where $h(q,d)$ is a scoring function that depends on a single query-document pair (q,d) and $s(q,q')$ represents the similarity between query q and q'. Assume that we can extract a feature vector x_{qd}, $x_{qq'}$ for each query-document pair (q,d) and each query pair (q,q'), the ranking function $f(q,d)$ described above can expressed as follows:

$$f(q,d) = h(x_{qd}) + \frac{1}{|S_q|}\sum_{q' \in S_q} s(x_{qq'})h(x_{q'd}),$$

where S_q is the set of candidate queries for the original tail query q. In the above expression of the ranking function $f(q,d)$, the first term $h(q,d)$ represents the degree of match between the original tail query q and the document d. On the other hand, the second term captures the contribution from a set S_q of other queries, weighted by their similarity $s(q,q')$ to the original query q'. Intuitively, consider a tail query q, it can be quite difficult for the feature vector x_{qd} to capture the relevance between q and d since the click-through data associated with the query q is too sparse. Hence, it is difficult to obtain a good ranking function based only on the feature vector x_{qd}. However, if we can obtain $h(q',d)$ for some similar queries q' which has a lot of click-through data so that the relevance of the document d with respect to query q' can be estimated accurately, we can overcome the sparsity problem by utilizing the relevance scores $h(q',d)$ as well as the similarity between q and q'.

It turns out that we can estimate $h(x_{qd})$ and $s(x_{qq'})$ in a similar manner using gradient boosting [14]. To this end, we apply the squared error loss $L(y_{qd}, f(q,d)) = (y_{qd} - f(q,d))^2$, for example, to measure the degree of divergence between the training data and the model estimation. Thus, the objective function for the learning process can be expressed by minimizing the empirical risk on training data:

$$\min_{h,s} \mathcal{R}_{reg}(f) = \sum_{q,d} (y_{qd} - f(q,d))^2,$$

which is equivalent to

$$\mathcal{R}_{reg}(h,s) = \sum_{q,d} \left(y_{qd} - h(x_{qd}) - \frac{1}{|S_q|} \sum_{q' \in S_q} s(x_{qq'}) h(x_{qd}) \right)^2. \quad (1)$$

The learning process for the pair-wise loss are quite similar, which implies the following loss function

$$L(f(x_{qd}), f(x_{qd'})) = \max(0, f(x_{qd'}) - f(x_{qd}) + \tau)^2, \quad (2)$$

for all pairs of documents associated to the same query such that $y_{qd} > y_{qd'}$ [14].

4.2 Alternating Minimization

To optimize the objective functions described above, we can minimize the empirical risk $\mathcal{R}(h,s)$ to obtain the two functions $h(x_{qd})$ and $s(x_{qq'})$ required to calculate the relevant scores. For the sake of concreteness, we describe the algorithm to optimize the objective function defined Equation (1) in detail. The pairwise loss function can be optimized in a similar approach. To this end, we apply the idea of alternating minimization. Specifically, we perform the following two steps alternatively:

- Given $s(x_{qq'})$, we can optimize $\mathcal{R}(h,s)$ with respect to h by

$$\min_{h \in \mathcal{H}_h} \sum_{q,d} \left(y_{qd} - h(x_{qd}) - \frac{1}{|S|} \sum_{q' \in S} s(x_{qq'}) h(x_{qd}) \right)^2$$

- Given $h(x_{qd})$, we can optimize $\mathcal{R}(h,s)$ with respect to s by

$$\min_{s \in \mathcal{H}_s} \sum_{q,d} \left(y_{qd} - h(x_{qd}) - \frac{1}{|S_q|} \sum_{q' \in S_q} s(x_{qq'}) h(x_{qd}) \right)^2$$

The empirical risk is ensured to decrease as we iterates the above two steps alternatively. Thus, we can obtain an estimation of score function h and similarity function s that is a minimal point of $\mathcal{R}$. We apply gradient boosting to estimate both functions in this study.

4.3 Feature Extraction

The proposed framework models two functions $h(x_{qd})$ and $s(x_{qq'})$ depending on two different sets of features: *query-document features* and *query-similarity features*. The query-document features x_{qd} capture the relevance between a pair of query q and document d. These features are widely applied in today's search engines to determine the relevance of documents.

In order to construct the similarity function $s(x_{qq'})$ in our framework, we also need to extract query-similarity features for each pair of queries. These features capture the degree of similarity between two queries. To this end, we construct a high-dimensional feature vector for each pair of queries and other features can be included naturally to improve the performance. In general, we avoid features based on user behavior which are very rare for tail queries. The extracted features can be categorized into three types: string matching, term matching and semantic matching.

5. EXPERIMENTS

We make use of the normalized discount cumulative gain (NDCG)to evaluate the performance of the proposed method, which is defined as follows:

$$\text{NDCG@n} \quad = \quad Z_n \sum_{i=1}^{n} \frac{2^{r_i} - 1}{\log(i+1)},$$

where r_i is the grade assigned to the i-th document of the ranking list. The term $\log(i+1)$ takes the position of ranking list into account by assigning larger weight to the top of the ranking list. The constant Z_n is chosen so that the perfect ranking gives an NDCG value of 1.

5.1 Experiments on the Microsoft Learning to Rank Data

We first describe experiments conducted over the Microsoft Learning to Rank data sets[1]. Our main purpose is to evaluate and verify that the proposed framework can utilize similar queries to enhance the quality of the learned ranking functions. There are two data sets Web10K and Web30K containing 10,000 and 30,000 queries respectively.

In order to evaluate the proposed framework, we need to extract features for a given pair of queries. However, the terms in queries are not available for this data set. Thus, we preprocess the data set as follows to *simulate* features for measuring query similarities: 1) We average the query-document features for all documents associated with each query and use the averaged features as the query features x_q for the corresponding query. 2) For each of the original query q, we generate a set of similar queries q'. In order to simulate similar queries q', we generate a random vector $p_{q'}$ by sampling each component uniformly from $[0,1]$. The query-document features $x_{q'd}$ is simulated by multiply each component of the random vector $p_{q'}$ and the query-document features x_{qd} for the original query. 3) The query-similarity features $x_{qq'}$ are generated by multiply each component of x_q and $p_{q'}$.

We sample 1000 queries from Web10K and Web30K as our training data. The performance is measured by the averaged NDCG over five independent samples. We compared CollRank to two baselines described as follows: GBTree: The ranking function is trained with gradient boosting using only the original queries and documents. AvgFeature: In this baseline, we extend the feature space by utilizing similar queries. We first average the feature vectors $x_{q'd}$ of similar queries and then append the averaged feature vector to the original feature vector x_{qd}. For each of the above method, gradient boosting is applied to train models using the extended feature vectors. We apply two different loss functions to learn the ranking function. The methods with squared and pair-wise loss functions are labeled with suffix Reg and Pair, respectively. For example, the method CollRank-Reg

[1] http://research.microsoft.com/en_us/projects/mslr/

represents the CollRank method with the squared loss defined in Equation (1). The CollRank-Pair represents the CollRank method with the pair-wise loss function defined in Equation (2).

From Figure 3, we can see that CollRank outperforms GBTree on both data sets, which indicates that CollRank can make use of similarity queries to enhance the ranking function. Moreover, AvgFeature outperforms GBTree which suggests that extending feature space can be quite useful to improve the quality of ranking. We can also observe that the pairwise loss function performs better than squared loss, which is expected because the pairwise loss function emphasises on the relative orders of the documents rather then their absolute scores.

5.2 Experiments on a Commercial Search Engine Data

5.2.1 Data Set Description

The data set is collected by a commercial search engine by sampling a number of queries from the query log. About 2000 features are used for each query-document pair. Five-level relevance judgements are associated with the query-document pairs. These judgments are utilized to train and evaluate the quality of the learned ranking functions. In order to evaluate the proposed method, we randomly split the data into training and test sets according to queries. For better analysis of the results, we further partition the test set into two subsets Test1 and Test2 according to the frequency of queries in query log. In particular, the set Test2 contains queries that are observed for the first time by the search engine.

Similar Query Set Generation. In order to construct a similar query set S_q of queries for each query q, we first look up its documents in the click-through data and aggregate queries associated with these documents. We generate the set S_q for a given query q by randomly sampling k different queries from this set of queries. In general, the size of the candidate set k is set to be 5 in our experiments.

Baselines. We compare the proposed method with the following baselines obtained by gradient boosting trees which is the state-of-the-art method for learning to rank: **NoClick.** In this method, we ignore the click-through data for all queries in this method and construct ranking functions with gradient boosting trees. **ClickStream.** In this method, all the clicked queries associated with a documents are aggregated and viewed as meta text contents of the documents called click stream. All content based features can be applied to this click stream. Therefore, in this method, the features extracted from the click stream is used together with other features in NoClick to construct ranking functions with gradient boosting trees. **SimLabel.** A straight forward method to utilize similar queries is to make use of them to expand training set. Specifically, assume that we have the labeled data (q, d, y_{qd}) in the training set. We can add a new training data (q', d, y_{qd}) if the query q and q' are quite similar to each other. Then, we include new training data in the training set and construct ranking function with gradient boosting trees. In this method, we measure the similarity between queries by the cosine similarity of the query feature vector and add a new training example when the similarity is greater than 0.7. **SimFeature.** In this baseline, we aggregate features from similar queries and append it to the original features of the query-document pairs. Specifically, the aggregated features are obtained by the weighted average of feature vectors corresponds to similar queries and documents. Formally, we have $x'_{qd} = \frac{\sum_{q' \in S_q} s_{qq'} x_{q'd}}{\sum_{q' \in S_q} s_{qq'}}$. The number of similar queries is set to be 5 in our evaluations.

For each of the above method, we apply two different loss functions to learn the ranking function. The methods with squared and pair-wise loss functions are labeled with suffix Reg and Pair, respectively.

We construct ranking functions with all the five methods on the training set and evaluate the obtained ranking functions on the test sets.

First, we can observe that ClickStream outperforms NoClick consistently, which indicates that click-through data can be quite important to improve the relevance of rankings. In fact, ClickStream extracts query-document features from the aggregated click logs, which is a common practice in Web search engines. In SimFeature, SimLabel and CollRank, query-document features extracted from click streams are also used. Comparing the performance over Test1 and Test2, we can observe that the performance measured by NDCG is lower on Test2 than on Test1. This is expected since Test2 contains queries that occur only once to the search engine. Thus, the problems with tail queries are much more significant over this test set than Test1.

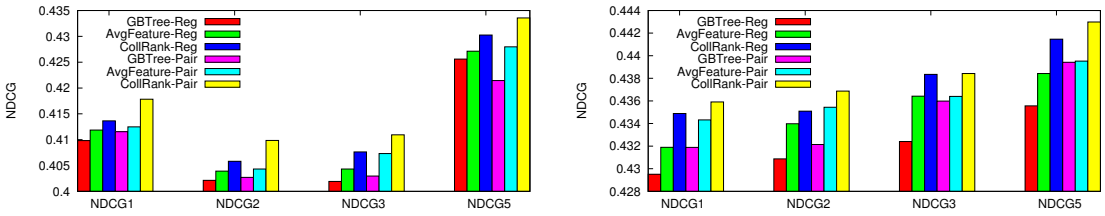
We can observe that CollRank with squared and pair-wise loss functions outperform the corresponding baselines. Moreover, the improvements are *significant* according to t-test over queries with significance level $p = 0.05$, which indicates that CollRank can better utilize the information from similar queries to improve the relevance of ranking.

In addition, we can see SimFeature actually performs quite well: it outperforms the other three baselines. We think this is because that the quality of features is the most important factor that impacts the performance of tail queries as we analyzed in Section 3. By making use of features from similar queries, SimFeature can overcome the sparsity problem associated with tail queries and thus improve the performance. From this viewpoint, the proposed CollRank also utilizes the features from similar queries to calculate the ranking scores of a given query-document pair, which partially explains the reason why CollRank can improve the relevance of tail queries. On the other hand, SimLabel does not perform as well as other methods. One reason is that SimLabel makes clear-cut decision of adding extra data rather than a soft-weighting, and the labeled data introduced by SimLabel can be quite noisy and thus reduce the performance of the obtained ranking functions.

6. CONCLUDING REMARKS

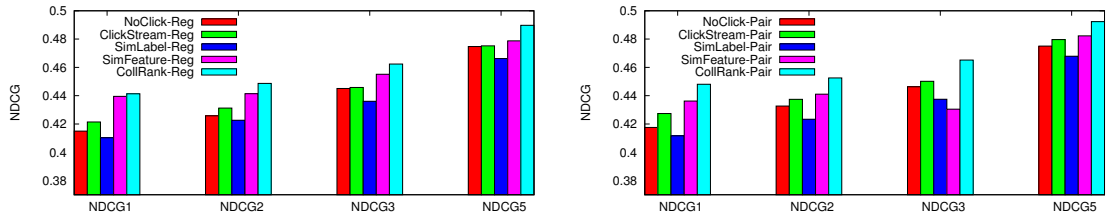
In this work, we address the problem of improving the relevance of tail queries and propose CollRank, a framework that utilizes the similarities between queries to overcome the data sparsity problem associated with tail queries. In particular, the similarity function between queries and the ranking function are estimated jointly from the training data so that they can adapt to each other well based on the given training data. An alternating minimization process is applied to optimize the objective functions. Extensive evaluations conducted over two data sets show that the proposed method can improve the relevance of ranking by making use of similar queries.

For future work, we plan to investigate other information



(a) Performance on Web10K (b) Performance on Web30K

Figure 3: The performance comparisons of CollRank and baselines on MSLR data set



(a) Performance on Test1 with square loss (b) Performance on Test1 with pair-wise loss

(c) Performance on Test2 with square loss (d) Performance on Test2 with pair-wise loss

Figure 4: The performance comparisons of CollRank and four baseline methods

sources to further improve the ranking of tail queries. For example, we can incorporate open knowledge base such as Wikipedia into our framework.

7. ACKNOWLEDGEMENT

Part of the work is supported by NSF IIS-1116886, NSF IIS-1049694, NSFC 61129001/F010403 and the 111 project.

8. REFERENCES

[1] I. Bordino, C. Castillo, D. Donato, and A. Gionis. Query similarity by projecting the query-flow graph. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522. ACM, 2010.

[2] F. De Bona, S. Riezler, K. Hall, M. Ciaramita, A. HerdaÇğdelen, and M. Holmqvist. Learning dense models of query similarity from user click logs. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 474–482. Association for Computational Linguistics, 2010.

[3] D. Downey, S. Dumais, and E. Horvitz. Heads and tails: Studies of web search with common and rare queries. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 847–848. ACM, 2007.

[4] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 170–178, San Francisco, CA, USA, 1998.

[5] A. Hust. Learning Similarities for Collaborative Information Retrieval. In *Proceedings of KI 2004 workshop Machine Learning and Interaction for Text-Based Information Retrieval*, volume 4, pages 43–54, 2004.

[6] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 133–142, New York, NY, USA, 2002.

[7] D. Metzler, S. Dumais, and C. Meek. Similarity measures for short segments of text. *Advances in Information Retrieval*, pages 16–27, 2007.

[8] S. Pandey and M. Fontoura. Estimating Advertisability of Tail Queries for Sponsored Search. *Annual ACM Conference on Research and Development in Information Retrieval*, pages 563–570, 2010.

[9] D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. LambdaMerge: merging the results of query reformulations. In *Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11*, page 795, New York, New York, USA, 2011. ACM Press.

[10] Y. Song and L.-w. He. Optimal rare query suggestion with implicit user feedback. *Proceedings of the 19th international conference on World wide web - WWW '10*, page 901, 2010.

[11] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th international conference on World wide web*, pages 47–56. ACM, 2011.

[12] J. Xu and G. Xu. Learning similarity function for rare queries. In *Proceedings of the fourth ACM international conference on Web search and data mining*, number 49, pages 615–624. ACM, 2011.

[13] H. Zaragoza, B. B. Cambazoglu, and R. Baeza-Yates. Web search solved? In *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, CIKM '10, page 529, New York, New York, USA, 2010. ACM Press.

[14] Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR 2007: Proceedings of the 30th Annual ACM Conference on Research and Development in Information Retrieval*, 2007.