To Divide and Conquer Search Ranking by Learning Query Difficulty

Zeyuan Allen Zhu[12*], Weizhu Chen[2], Tao Wan[3], Chenguang Zhu[24], Gang Wang[2], Zheng Chen[2]

[1]Fundamental Science Class, Department of Physics, Tsinghua University, Beijing, China, 100084
zhuzeyuan@hotmail.com

[2]Microsoft Research Asia, Beijing, China, 100080
{v-zezhu, wzchen,v-chezhu, gawa, zhengc}@microsoft.com

[3]School of Computer Science and Technology, Tianjin University, Tianjin, China. 300072.
taowan.wtommy@gmail.com

[4]Department of Computer Science and Technology, Tsinghua University, Beijing, China. 100084.
zcg.cs60@gmail.com

ABSTRACT
Learning to rank plays an important role in information retrieval. In most of the existing solutions for learning to rank, all the queries with their returned search results are learnt and ranked with a single model. In this paper, we demonstrate that it is highly beneficial to divide queries into multiple groups and conquer search ranking based on query difficulty. To this end, we propose a method which first characterizes a query using a variety of features extracted from user search behavior, such as the click entropy, the query reformulation probability. Next, a classification model is built on these extracted features to assign a score to represent how difficult a query is. Based on this score, our method automatically divides queries into groups, and trains a specific ranking model for each group to conquer search ranking. Experimental results on RankSVM and RankNet with a large-scale evaluation dataset show that the proposed method can achieve significant improvement in the task of web search ranking.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms
Algorithms, Performance, Theory

Keywords
Learning to Rank, Query Difficulty.

1. INTRODUCTION
Search users currently pay more and more attention to search engines' relevance since it is a challenge task to rank the Web-scale WebPages successfully for various queries. Learning to rank is one of the techniques used to improve the relevance of search ranking. By leveraging machine learning techniques into model construction, learning to rank is known to build the ranking model theoretically and practically effectively [1] [7] [3]. Recently, many learning-based methods have been proposed to employ different strategies in learning to rank, such as RankSVM [2] [7], RankNet [1] and Rankboost [3], and show their effectiveness to improve search relevance.

CIKM'09, November 2–6, 2009, Hong Kong, China.

In most of the previous works in learning to rank, all the queries with their returned search results are learnt and ranked using a single ranking model. However, as found by [9], using one ranking model for all the queries is not an optimal solution to conquer the search ranking challenge. As is also found by [8], a good ranking algorithm for the topic relevance task does not always perform well for the homepage finding task. [11] observed that there are a lot of variations across queries. Inspired by these observations, if queries could be divided properly into different groups, it is possible to design different ranking models for each group to improve the search ranking.

However, dividing queries for learning to rank is not a trivial task. An intuitive way to divide queries is based on search intent, as discussed in [8], which used the query content information, URL and the link information from the retrieval results to classify queries into two categories, the topic query and the homepage-finding query. Another intuitive way to divide queries is based on the query length. [11] had shown that the query length is a key feature to distinguish query ambiguity. Intuitively, for long queries, a learning model is supposed to include more features to build a more complicated ranking function than short queries, like the query proximity in [10].

In this paper, we propose a method to characterize a query using a variety of features extracted from the large-scale user behavior in a commercial search engine, such as the click entropy, the query reformulation probability and the session click number. Through investigating all of these features, our method formalizes the query difficulty prediction as a binary classification problem. The output value of the classifier is defined as the query difficulty score in this paper. According to this score, we explore several strategies to divide queries into multiple groups and learn the ranking models separately. We conduct experiments on a large-scale dataset using RankSVM and RankNet as the separate ranking model. Experimental results show that our basic binary strategy, which divides queries into difficult and easy categories, can improve the Mean-NDCG value by 2.5% and 1.1% in RankSVM and RankNet respectively, where Mean-NDCG is the mean value of NDCG@1 through NDCG@10. Our cluster-based strategy, which divides queries into multiple groups based on the query difficulty score, can improve Mean-NDCG by 4.1% using RankSVM.

[*] This work was done when the first author was visiting Microsoft Research Asia.

To summarize, the contributions of this paper include:

1) We propose a method to measure query difficulty through using large-scale user search behavior and utilize it to divide and conquer learning to rank and achieve significant improvement in the task of search ranking.
2) We explore several strategies to divide queries to conquer learning to rank and propose a cluster-based strategy which could improve the Mean-NDCG by 4.1%.
3) The proposed method is validated on the large-scale dataset with both RankSVM and RankNet learning algorithms. Experimental results demonstrate that it can consistently achieve significant improvements.

The rest of the paper is organized as follows. Section 2 describes our method to predict query difficulty. In Section 3, we present our method to divide and conquer learning to rank. Experimental results with discussion are presented in Section 4. And Section 5 concludes the paper with future work.

2. LEARNING QUERY DIFFICULTY

Query difficulty represents how difficult a query is for a ranking algorithm to return the satisfactory results to users. Generally, if the search results for a query are bad and users are not satisfied with them, we will regard it as a difficult query; vice versa. In this paper, we use large-scale human judged data to identify whether a query is a difficult query or an easy query.

We consider a wide range of features to characterize a query. The description of each feature and their explanations are presented in Table 1. We drilled down the features according to whether it is a Query feature or a Click feature, and whether it needs the session information to aggregate the feature value, where Session value with Yes means it needs session information to construct the value for this feature and Session No represents the opposite. For each query, there are some aggregation features in the Click type with Session, like the *click distribution at K* for query q, denoted as $P_{cd}(q,K)$, and the *click entropy*, which is a feature used in work [11] to measure the consistency for the clicked pages given a query q. We define their formulas as the followings:

$$P_{cd}(q,K) = \frac{Number\ of\ click\ at\ position\ K}{Number\ of\ all\ click}$$

$$Click\ Entropy(q) = -\sum_{K=0}^{10} P_{cd}(q,K) * \log_2(P_{cd}(q,K))$$

We further explain some other features in Table 1, *Probability of Reformation* is the fraction of reforming a new query to replace the current query; *Query Suggestion Probability* is the fraction that the current query generated by clicking query suggestion other than inputting to the search box; *Probability of Clicking Ads* is the fraction of advertisement click in the search click.

However, the human judged efforts can only cover a small range of queries and leave a large range of queries unlabeled. Therefore, we formalize the query difficulty prediction as a binary classification problem. To build a classifier to distinguish difficult queries from easy queries, each human judgment is used as a label, and a variety of features extracted from the user behavior log is used as the feature vector. For an unlabeled query, given the query feature vector as the input, the output value of the classifier is defined as the query difficulty score indicating how difficult a query is. In this paper, we select Support Vector Machine (SVM) as our classification algorithm. Given a query q, we define its query difficulty score $QD(q) = \langle w, \emptyset(q) \rangle + b$, where w is the SVM separation hyper-plane and $\emptyset(q)$ represents the feature vector of query q.

Table 1. Extracted features used to predict query difficulty

		Session	
		YES	NO
Features	Query	Probability of Reformulation Avg. Dwelling time Is the first query in session Is the last query in session Fraction of word shared with previous query Fraction of word shared with next query	Query length Number of characters Number of terms Capital letter in Query Is a typo Search result number
	Click	Click distribution At K Entropy of Click Distribution Average Click per Session Query Suggestion Probability Avg. click position	Avg. number of Click Avg. number of Click per User Probability of Click Probability of Clicking Ads

3. TO DIVIDE AND CONQUER LEARNING TO RANK

In computer science, divide and conquer (D&C) is an important strategy based on multi-branched recursion, which works by breaking down a problem into multiple sub-problems of the same type, until it becomes simple enough to be solved directly. When we apply the D&C strategy into the problem of learning to rank, the first action we are facing is how to divide the query effectively and conquer learning to rank by achieving better search relevance.

An intuitive way to divide queries is based on the length of the query. [11] showed that the query length is a key feature to distinguish query ambiguity. Intuitively, for long queries, learning model is supposed to include more features to build a more complicated ranking function than short queries, like query proximity in [10]. Hence, in the following experiments, we use the query length dividing strategy as a baseline in the comparative study, and referred it as the *length-based strategy*.

As is stated in [11], if a system could know which queries are hard, it can devote the appropriate resources to improve the results for those queries. In the task of learning to rank, if we could get a highly confident score indicating how difficult a query is, we can train different ranking models for different queries to conquer the ranking problem.

Based on the query difficulty score $QD(q)$ discussed in the previous section, we divide all the queries into two categories – the difficult queries and the easy queries, depending on whether $QD(q) > 0$. In the training process, we train a separate ranking model for either kind of query. In the testing process, we first use SVM to classify a query. Depending on whether it is a difficult query, the corresponding ranking model will be used for the prediction. We call this method the *Difficult&Easy strategy*.

Although the Difficult&Easy strategy seems to be a simple and practical solution to divide queries, it is possible to investigate a finer dividing strategy. Indeed, the difficulty score that presents the confidence of the classification [6] motivates us to divide queries into multiple categories. Therefore, an unsupervised dividing strategy is applied here using a clustering algorithm, Kmeans, performed on the output value of the classification function, which is $QD(q)$ in the previous section.

The reason that motivates us to choose Kmeans as the clustering algorithm is that K, the number of clusters, is a free parameter, enabling us to actively decide the number of query clusters. For each cluster, a separate model is trained. In the testing process, each query is first put into a cluster by classifying it to the closest centroid, the corresponding ranking model is then retrieved to generate the ranking result. We call this strategy the *Pre-cluster strategy*.

Throughout this paper, two state-of-the-art algorithms RankSVM [7] and RankNet [1] are to be used as the ranking model. We will also use their performance on the entire dataset as the baselines to compare with the divide and conquer technique.

4. EXPERIMENTAL RESULTS

4.1 Dataset

To learn the query difficulty score and conquer search ranking, we use two datasets obtained from a commercial search engine: the explicit dataset and the implicit dataset, both containing the same set of queries. A total of 5,676 queries were sampled from the log of a commercial search engine according to the query frequency.

In the explicit dataset, top 50 search results for each query are labeled by five assessors to represent the relevance score between the query and each result, ranging from 0 to 5. There are a total of 283,800 query-URL pairs. In order to conduct learning to rank on the realistic dataset, we pulled the Meta features from the search engine, which are used to predict ranking in the realistic search engine. A total of 200 features are incorporated.

For the implicit dataset, we retrieve the user behavior data from Sep. 1st 2008 to Sep. 30th 2008. There are a total of 46,392,929 user sessions, around 52,000,000 URLs and around 15,800,000 distinct users. For each query, a set of features is extracted from this dataset to build a feature vector according to Table 1.

Note that we did not use the public dataset like LETOR or OHSUMED, because there is no user behavior data for most of the queries in them and the number of queries in them is too small to perform persuasive experiments in our problem.

4.2 Query Difficulty Prediction

To generate the ground-truth training data to build a binary classier for query difficulty prediction, we leverage the explicit dataset to distinguish difficult queries from easy ones. As stated in Section 2, query difficulty could be used to measure whether a search engine could return satisfactory and relevant results to users. With all the query-URL relevance values and the realistic ranking positions in the explicit dataset, we can measure the query difficulty based on a calculation metric, like Precision, DCG and NDCG in [5]. However, which metric could perform the best for query difficulty prediction is an open question. We conduct a classification experiment to seek for answers.

We retrieve 1,000 queries from the explicit dataset. For the splitting between training and testing, we leverage the 5-fold splitting method in work [6] and average the classification accuracy [6] of each fold. The results for SVM with linear model are presented in Figure 1.

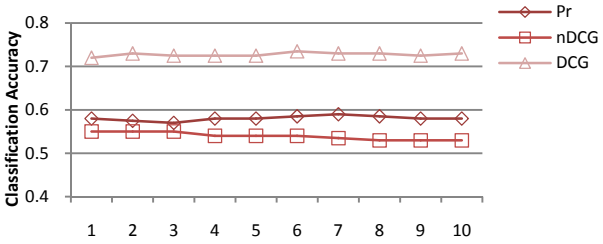

Figure 1: Classification accuracy for Pr/nDCG/DCG at different position, using linear SVM.

From Figure 1 we can see that the prediction with DCG performs significantly better than that with Precision and NDCG, partially due to the fact that DCG depicts the difficulty based on an ideal ranking. In the following experiments, we will use DCG@10 as the metric since 10 is the search result number for most search engines.

In Table 2, we demonstrate the feature weights generated by the classifier. The larger weight indicates the more reasonable this feature can explain the query difficulty. We can see from the table that our created features in Section 2 play a substantial role in explaining the query difficulty.

Table 2: Linear SVM features with its weights

weight	feature
0.698	Click Entropy
0.584	Reformation probability
0.574	Avg. click position
0.553	Query length
… …	… …
-0.285	Click Distribution @ 2
-0.709	Avg. Click per Session
-0.829	Click Distribution @ 1
-0.980	Avg. number of click per user

Based on the query relevance score calculated with the best metric, DCG, we sort all the 5,676 queries based on their DCG scores and split them into 2 subsets of equal size. The set with lower DCG is labeled as the difficult query set and the other is labeled as the easy set. Based on this data, the SVM classifier is learned to predict query difficulty scores to be used in the next sub-section.

4.3 Experimental Results on Dividing and Conquering Learning to Rank

Now we verify whether the divide and conquer strategy for learning to rank is well-justified. For all dividing strategies in Section 3, we use the 5-fold strategy to split training and testing sets to make sure every query has an opportunity to be evaluated. The experimental results reported are the average of each fold.

We use RankSVM and RankNet as base models. The parameters for RankSVM are set to default. The number of epoch in RankNet is set to 200 as default. Since we want to evaluate how the divide and conquer strategy performs on learning to rank, we use its opposite, a single model strategy, as the baseline, referred as *single* in figures. Since dividing by length is an intuitive strategy, *length-based strategy* is used as the other baseline.

4.3.1 Difficult&Easy Strategy

To verify the effectiveness of the Difficulty&Easy (D&E) strategy, we compare its NDCG and MAP scores with those of the length-based strategy and the single model strategy. We use RankSVM and RankNet as the base ranking models and present the results in Figure 2 and Figure 3. In Figure 2, the length-based strategy and the Difficulty&Easy strategy can achieve improvements of 1.9% and 2.5% respectively in the Mean-NDCG. These results justify that the dividing strategy could bring improvements in learning to rank. Furthermore, the Difficult&Easy strategy performs much better than length-based strategy, indicating that employing more features beyond length to estimate the query difficulty can benefit learning to rank.

In Figure 3, an informative observation is that the length-based strategy could not achieve a better performance than the single model strategy. However, the Difficult&Easy strategy can still bring improvements. The Mean-NDCG improvement in RankNet is 1.1%. Although the improvement of RankNet is a little smaller than that of RankSVM, it still draws the same conclusion as RankSVM. Thus, we will only use RankSVM in the following experiments due to space consideration.

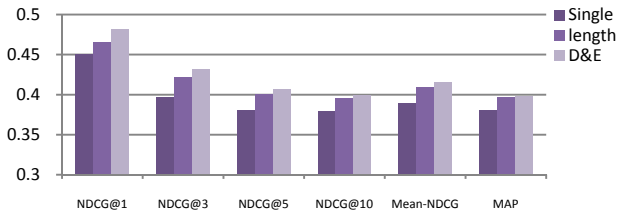

Figure 2: Ranking performance with RankSVM as base model

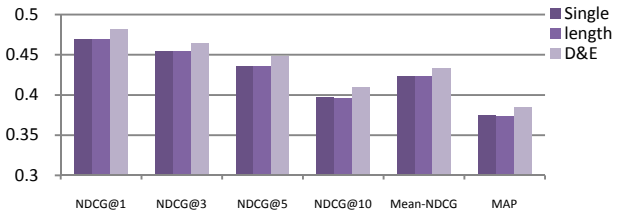

Figure 3: Ranking performance with RankNet as base model

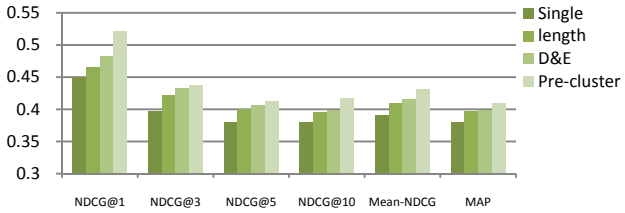

Figure 4: Ranking performance after adding Pre-cluster

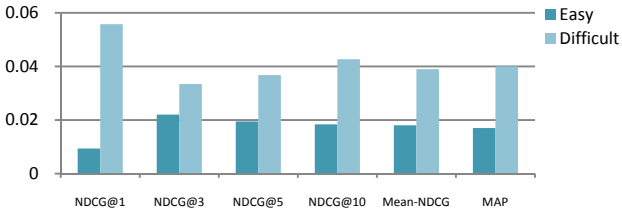

Figure 5: NDCG improvement for Easy and Difficult Queries

4.3.2 Pre-cluster Strategy

The only parameter in the Kmeans algorithm is K, the number of clusters. We divide the first fold data into two subsets where 4/5 of the data is used as training, and the other 1/5 data is used to tune a K value with the best NDCG. Based on our experiment, we use K=17 in the pre-cluster strategy.

In Figure 4, we can see that Pre-cluster strategy brings significant improvements on all NDCG positions and the MAP measurement. It outperforms the single model strategy, the length-based strategy, and the Difficult&Easy strategy by 4.1%, 2.2% and 1.5% respectively on the Mean-NDCG. We continue to conduct the t-test on the improvements with results indicating the improvements are statistically significant (p-value<0.05). When compared with the Difficult&Easy strategy, the 1.5% improvement verifies that using a cluster strategy on the difficulty score is more successful than the strategy with two categories.

4.3.3 Discussion

We have achieved significant improvement over the single model baseline, and a continuing investigation could be which kind of query plays a more substantial role in obtaining the improvement. To answer this question, we calculate the average improvement delta between the single model baseline and the pre-cluster strategy, for both the difficult query set and the easy query set.

Experimental results from Figure 5 are informative, in which we can see that difficult queries have much more improvements than easy queries for all the NDCG positions and MAP. We hence conclude that difficult queries have more potential to be improved and need to be handled separately to conquer learning to rank.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a method to measure query difficulty and utilize it to divide and conquer learning to rank. In contrast to existing methods, we leverage the explicit and implicit user data to predict query. Experimental results on a large-scale realistic dataset show that the dividing and conquering strategy based on the query difficulty score can achieve significant improvements when compared with the single model baseline. Among all the dividing strategies, the Pre-cluster strategy on query difficulty score performs the best with a 4.1% improvement on the Mean-NDCG, when compared with the baseline.

There are several aspects in this direction that we can pursue to further enhance our method. For query difficulty prediction, we plan to investigate other nonlinear solutions for example using rbf kernels. For the dividing and conquering strategies, we hope to investigate more dividing strategies, such as by navigational and informational, and see the possibility to perform a comparable study for different kinds of dividing strategies.

6. REFERENCE

[1] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to Rank using Gradient Descent. In *ICML* (2005).

[2] Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., and Hon, H.-W. Adapting ranking SVM to document retrieval. In *SIGIR* (2006).

[3] Freund, Y., Schapire, R. E., and Singer, Y. An efficient boosting algorithm for combing preferences. *JMLR*, 4 (2003).

[4] Geng, X., Liu, T.-Y., Qin, T., Arnold, A., Li, H., and Shum, H.-Y. Query dependent ranking using K-nearest neighbor. In *SIGIR* (2008).

[5] Järvelin, K. and Kekäläinen, J. IR evaluation methods for retrieving highly relevant documents. In *SIGIR* (2000).

[6] Joachims, T. *Learning to Classify Text Using Support Vector Machines*. Kluwer Academic Publisher, 2002.

[7] Joachims, T. Optimizing search engines using clickthrough data. In *SIGKDD* (2002).

[8] Kang, I. and Kim, G. Query type classification for Web document retrieval. In *SIGIR* (2003).

[9] Qin, T., Zhang, X.-D., Wang, D.-S., Liu, T.-Y., Lai, W., and Li, Hang. Ranking with Multiple Hyperplanes. In *SIGIR* (2007).

[10] Tao, T. and Zhai, C. An exploration of proximity measures in information retrieval. In *SIGIR* (2007).

[11] Teevan, J., Dumais, S. T., and Liebling, D. J. To Personalize or Not to Personalize: Modeling Queries with Variation in User Intent. In *SIGIR* (2008).