Northeastern University in TREC 2009

Million Query Track

Evangelos Kanoulas[‡], Keshi Dai[*], Virgil Pavlu[*], Stefan Savev[*], Javed Aslam[*]

[‡] Information Studies Department, University of Sheffield, Sheffield, UK
[*] College of Computer and Information Science, Northeastern University, Boston, MA, USA

1 Introduction

Ranking is a central problem in information retrieval. Modern search engines, especially those designed for the World Wide Web, commonly analyze and combine hundreds of features extracted from the submitted query and underlying documents in order to assess the relative relevance of a document to a given query and thus rank the underlying collection. The sheer size of this problem has led to the development of *learning-to-rank* (LTR) algorithms that can automate the construction of such ranking functions: Given a training set of (feature vector, relevance) pairs, a machine learning procedure learns how to combine the query and document features in such a way so as to effectively assess the relevance of any document to any query and thus rank a collection in response to a user input.

Much thought and research has been placed on the development of sophisticated learning-to-rank algorithms. However, relatively little research has been conducted on the construction of appropriate learning-to-rank data sets nor on the effect of these data sets on the ability of a learning-to-rank algorithm to "learn" effectively.

Given that the IR technology is ubiquitous in a vast variety of contexts and environments it is not unreasonable to assume that searchable material (corpora) and user information needs will radically vary from one retrieval environment to another. Theoretically, ranking functions should be trained over collections with similar characteristics as the collections they will be deployed in. However, the ability to construct different ranking functions for different retrieval environments is limited by the cost of constructing such customized training collections. Thus, the question that naturally arises is whether training on a collection of certain characteristics can still lead to an effective ranking function over collections of different characteristics. To answer this question we trained our ranking functions (by employing SVM) over two different collections, (a) the Million Query 2008 (MQ08) collection (GOV2 corpus and queries with at least one click on documents in the .gov domain), and (b) a Bing generated collection (described in Section 2.1) and employed the constructed ranking function over the Million Query 2009 (MQ09) collection (ClueWeb09 corpus and general web queries).

Furthermore, even within a certain retrieval environment (represented by a given collection) different queries may have radically different characteristics and thus different features may better capture the notion of relevance. For instance, in the case of precision-oriented queries, such as homepage/namepage finding, the url of a document or its popularity may be more indicative of the document relevance than the document text itself, while for informational queries the document url maybe less indicative than its text. Most of the existing learning-to-rank approaches train a single ranking function to handle all queries. Hence, the question that arises is whether training a different ranking function for each one of these different query

categories and using the appropriate ranking function when a user submits a new query may lead to better retrieval performance than employing a single ranker for all queries.

Geng et al. [4] proposed query-dependent ranking. According to their approach when a user submits a query the K Nearest Neighbor (KNN) method is employed to identify the closest queries to the submitted one and a ranker is trained over these K queries. This query-dependent ranker is then used to rank documents with respect to the submitted query. The KNN query-dependent ranker model was then compared against the single ranker model (baseline) and a query classification ranker model. According to the latter model different rankers were constructed for homepage finding, namepage finding and topic distillation. The KNN approach slightly outperformed the other two (with an nDCG difference of approximately 0.01-0.02).

Web queries however may be classified based on a variety of different criteria, such as query hardness, user intent, ambiguity, popularity (frequency in a query log), spatial or temporal characteristics, length, click-through rates, verbosity (i.e. a natural language query versus a set of keywords). In this work we explore whether constructing different ranking functions for different query intent and query hardness leads to better retrieval results than constructing a single ranker for all queries. However, given that we trained over the MQ08 collection, our ability to answer this question highly depends on the robustness of the learning-to-rank algorithm when trained and tested over different collections (that is, it highly depends on the answer to the first question posed).

2 Methodology

We indexed the GOV2 collection and the ClueWeb09 (category B) collection using the Indri Search Engine from the Lemur Toolkit [1]. The constructed indexes contained five fields: document, title, heading, anchor text, and url; this allowed extracting features from all these different fields.

Based on the Indri indexes, we extracted a total of 57 LETOR4.0 -like features for each one of the query-document pairs. We did not utilize the available LETOR4.0 MQ08 training set [2] so that the extracted features from ClueWeb09 would be comparable to the ones from MQ08. A summary of the features can be seen in Table 1. Text features were extracted for all 5 fields.

The MQ08 data set (training data) consists of 784 queries from the Million Query 2008 collection. 403 queries had 8 documents labeled with relevance judgments, 204 queries had 16 documents labeled, 102 queries had 32 documents labeled, 50 queries had 64 documents labeled and 25 queries had 128 documents labeled. The corpus of the MQ08 collection is the GOV2 corpus and the queries in the collection had at least one click on documents in the .gov collection. Given the computational complexity of feature extraction and training features were extracted only from the union of (a) top 1,000 documents ranked by Indri language model [1] over the document text, (b) the top 500 documents ranked by Indri over the anchor text, and (c) the top 500 documents ranked by Indri over the url of the documents. A state-of-the-art SVM learning-to-rank algorithm [5] was employed to construct the ranking functions.

2.1 Training and testing over collections with different characteristics

Our first goal was to explore whether training over a collection with characteristics different than the collection the ranking function will be deployed in can still lead to effective retrieval.

We trained the SVM over the MQ08 data set and then used the resulting ranking function (**NeuSvmBase**) to re-rank the 2,000 documents (described above) per query in the new MQ09 collection.

There are two striking differences between the MQ08 and the MQ09 collections. (1) the MQ08 collection is on a .gov corpus with .gov related queries, while the MQ09 collection is a general Web collection,

Text Features	
T1.	length
T2.	TF
T3.	IDF
T4.	TF*IDF
T5.	normalized TF
T6.	Robertson's TF
T7.	Robertson's IDF
T8.	BM25
T9.	Language Model (Laplace Smoothing)
T10.	Language Model (Dirichlet Smoothing)
T11.	Language Model (JM Smoothing)

Web Features	
W1.	number of incoming links
W2.	number of incoming links from different domains

Table 1: Features extracted from the MQ08 (GOV2), the Bing generated and the MQ09 (ClueWeb09 category B) collections.

and (2) the MQ08 is a spam-free collection while the MQ09 is not.

In order to be able to separate the conflated effects of these two different characteristics of the two collections, we constructed a second ranker (**NeuSvmStefan**). We again used the SVM ranking algorithm but instead of training over the MQ08 collection we first obtained queries from the query log of a commercial search engine (different than the MQ09 queries), then we submitted these queries to Bing and finally trained over the intersection of the top 100 documents returned by Bing and the documents included in the ClueWeb09 category B crawl. Since there were no relevance judgments for the returned by Bing documents we used the reverse of the document ranks. This Bing-generated training collection has the same characteristics as the MQ09 collection except that it is (effectively) spam-free.

2.2 Query-dependent rankings

The second question we attempted to answer was whether constructing separate ranking function for different query categories can lead to more effective retrieval than constructing a single ranker. The query characteristics we used to categorize queries were query intent (i.e. precision-oriented queries vs. recall-oriented queries) and query hardness (i.e. hard vs. easy queries).

The training data set was the MQ08 data set. Regarding the query intent we manually classified the 784 queries in the MQ08 collection. The Average Average Precision (AAP) based on the Average Precision scores of the participating runs in the MQ08 track was utilized to classify queries into hard and easy.

Then, we trained three sets of ranking functions. The first set consisted of two rankers, one over the precision-oriented queries and one over the recall-oriented ones (**NeuSvmPR**). The second set consisted again of two rankers, one over the hard and one over the easy queries (**NeuSvmHE**), while the last set consisted of four rankers, a ranker for each one of the four combinations, i.e. precision-oriented and hard, recall-oriented and hard, etc. (**NeuSvmPRHE**).

	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	0.283	0.294	0.325	0.300	0.278

Table 2: 5-fold cross validation on all judged documents of MQ09 using regression

For MQ09 queries, in order to determine what ranker to use, we had to estimate the intent (precision vs. recall) and the difficulty (hard vs. easy):

- Precision vs. Recall: We constructed an SVM classifier, trained it over MQ08 queries using as features the average feature values of the relevant document for each query with the manual labels. We applied the classifier over the MQ09 queries using as features the average features of the top 100 documents (pseudo-relevant).
- Hard vs. Easy: We used the Jensen-Shannon distance of the document rankings generated by a number of ranking functions (e.g. BM25, Language Models etc.) on the MQ09 collection [3].

In both classifications a hard decision of whether a new query is precision-oriented or recall-oriented or whether it is hard or easy was made. Then the appropriate ranker was used to re-rank the top 2,000 documents originally ranked by the Indri language model as described above.

3 Results and Discussion

The baseline run of the single ranker trained over the Million Query 2008 collection (NeuSvmBase) resulted in an estimated Mean Average Precision (MAP) of 0.089, a particularly low score. To test the correctness of our SVM ranking algorithm and the correctness of the feature extractor we extracted features from the Million Query 2008 collection, and performed a five-fold cross validation. The retrieval performance achieved was at least as good as the LETOR 4.0 baselines. Thus, these results indicate that training over a collection of given characteristics cannot always lead to an effective ranking function when the function is deployed to rank documents in a collection of radically different characteristics.

Our second run of the single ranker trained over the Bing-generated collection (NeuSvmStefan) performed slightly worse than the baseline achieving an MAP score of 0.084. As mentioned earlier the Bing-generated training collection has similar characteristics to the Million Query 2009 collection except that it is spam-free. Therefore, a reasonable assumption is that training over a spam-free collection and using the ranker over a collection that includes spam leads to low retrieval effectiveness. We manually inspected the top documents of our baseline ranker and observed that although the original by the language model ranking included many relevant documents at the top positions the learning-to-rank algorithm boosted spam documents towards the top position of the re-ranked list. An interesting question that arises here is how much the effectiveness of our rankings could improve by simply removing all the returned spam documents. Further, a future direction of research would be to compare the retrieval effectiveness of rankers trained over collections that includes spam and rankers trained over spam-free collections with anti-spamming applied over the results of both rankers.

After obtaining the query categories and judgments form TREC MQ09, we trained and tested our ranking function over the MQ09 data by performed a 5-fold cross validation. When testing our ranking function we only considered the judged documents in a LETOR-like manner and thus the performance of our ranker is not comparable to the ones reported by TREC for our submitted rankers. However, the results for each fold can be viewed in Table 2 and the mean average precision achieved was 0.296.

Train	Test	MAP
Recall	Precision	0.37
Precision	Precision	0.38
Precision	Recall	0.49
Recall	Recall	0.52

Table 3: Training & Testing on different query categories over all judged documents of MQ09 using regression

	Easy	Medium	Hard
predicted Hard	115	91	63
predicted Easy	107	103	121

Table 4: Hard vs. Easy Queries

	Precision	Recall
predicted Precision	81	108
predicted Recall	176	235

Table 5: Precision vs Recall Queries

Given the failure of the baseline learning-to-rank algorithm to learn an effective for the Million Query 2009 collection ranking function, we could not answer our second question of whether query-dependent rankings over predefined query categories can lead to significant improvements when compared with a single ranker for all queries. The estimated MAP scores were all slightly lower than the baseline but any conclusions would be misleading.

However, we conducted a similar experiment over the MQ09 collection after the qrels and the query categories were released by the Track. We built training and testing data sets based on the different user intent (i.e. one precision-oriented query data set and one recall-oriented query data set). There were in a total of 176 precision-oriented and 230 recall-oriented queries labeled by NIST assessors. We trained two different ranking functions over the two training sets and tested the ranking functions both against the precision- and against the recall-oriented testing sets. The results (shown in Table 3) illustrate that a ranking function trained over precision-oriented queries outperforms a ranking function trained over recall-oriented queries when the testing set consists of precision-oriented queries only, while the opposite is true in the case of a recall-oriented query test set, as expected. A further observation is that recall-oriented queries appear to be more useful for training than precision-oriented ones, since the performance difference between the two rankers is large when the test set consists of recall-oriented queries but very small when it consists of precision-oriented ones. This however needs further investigation since the set of recall-oriented queries was much larger that the set of precision-oriented queries which could also have affected the effectiveness of the trained ranking functions.

Finally, the results of our classification can be viewed in Tables 4 and 5. The query intent classifier seemed to biased towards recall-oriented queries by classifying 411 out of 600 queries as recall-oriented. Out of those 411 predicted recall queries 235 where assessed as recall queries by the judges. The Jensen-Shannon methodology to classify queries based on their hardness does not seem to have done well either. Given that Jensen-Shannon has been shown to predict query hardness well when it is applied over rankings by TREC runs this may indicate that applying the same methodology over rankings by basic ranking functions (e.g. BM25, LM) does not lead to equally good predictions.

References

[1] The lemur toolkit. http://www.lemurproject.org.

[2] Letor 4.0. http://research.microsoft.com/en-us/um/beijing/projects/letor/.

[3] J. Aslam and V. Pavlu. Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions. *Lecture Notes in Computer Science*, 4425:198, 2007.

[4] X. Geng, T. Liu, T. Qin, A. Arnold, H. Li, and H. Shum. Query dependent ranking using k-nearest neighbor. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 115–122. ACM, 2008.

[5] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 226. ACM, 2006.