Incorporating Query-Specific Feedback into Learning-to-Rank Models

Ethem F. Can, W. Bruce Croft, R. Manmatha
Center for Intelligent Information Retrieval (CIIR)
School of Computer Science
UMass Amherst
efcan, croft, manmatha@cs.umass.edu

ABSTRACT

Relevance feedback has been shown to improve retrieval for a broad range of retrieval models. It is the most common way of adapting a retrieval model for a specific query. In this work, we expand this common way by focusing on an approach that enables us to do query-specific modification of a retrieval model for learning-to-rank problems. Our approach is based on using feedback documents in two ways: 1) to improve the retrieval model directly and 2) to identify a subset of training queries that are more predictive than others. Experiments with the Gov2 collection show that this approach can obtain statistically significant improvements over two baselines; learning-to-rank (SVM-rank) with no feedback and learning-to-rank with standard relevance feedback.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Experimentation

Keywords

Learning-to-rank, query-specific feedback, relevance feedback

1. INTRODUCTION

Consider a case where there are navigational and history related queries available in the training set. For a history related test query, a retrieval model learned using all of the queries in the training set might not be as good as a model learned using queries only related to history. Training queries that are similar to a test query can rank that particular test query better than others.

Relevance feedback methods have been studied and used for some time in information retrieval. Feedback is either explicit (i.e., where the user provides the relevance information for some retrieved items of the test query) or implicit (i.e., where the top ranked documents are assumed to be relevant). In both cases, the main idea is to use information from the initial search to improve retrieval performance. Relevance feedback is performed using the top retrieved documents to improve the retrieval by modifying the initial retrieval model. In this work we focus on situations where explicit feedback is available from the user for a few top documents. This typically happens when searching for patents or intelligence information where a user may be willing to spend additional time to label the documents from an initial search as relevant/non-relevant to improve the results for the rest of the retrievals. We consider a setting where documents in the top k are assessed, with the goal of improving the the other documents ranking.

In the learning-to-rank framework [9, 17], parameters for a retrieval model are learned based on training data consisting of queries and associated relevant and non-relevant documents. A common approach in learning-to-rank is to use machine learning techniques that optimize a set of weights on joint query-document features to maximize the number of pairs where a relevant document is ranked higher than a non-relevant document. Figure 1 illustrates the learning-to-rank framework. $Q_1, ..., Q_k$ are the queries in the training set and Q_t is the test query.

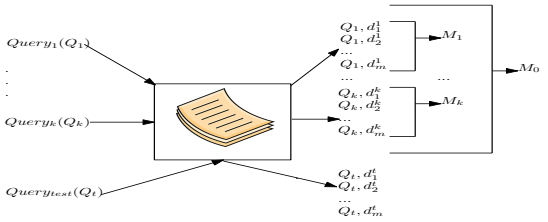

Figure 1: Illustration of learning-to-rank framework.

Typically a learning-to-rank approach estimates one retrieval model across all training queries $Q_1, ..., Q_k$ (represented by feature vectors), after which the test query (Q_t) is ranked upon the retrieval model and the output is presented to the user. We extend this approach by an additional step; we refer to the learning-to-rank model which is trained across all queries ($Q_1, ..., Q_k$) as the initial retrieval model (M_0) and the induced ranking for the test query as initial ranking. We target a situation where partial relevance assessments are available on the initial ranking, for example in the top 10. Our goal is to leverage this user feedback to improve the ranking of unjudged documents in context to the initial rank-

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.
ACM 978-1-4503-2257-7/14/07.
http://dx.doi.org/10.1145/2600428.2609503 .

ing. An obvious way of doing so is to learn a learning-to-rank model on the partial judgments M' and present a ranking of a combination of the initial model (M_0) and partial model (M') to the user. In this work, we evaluate an alternative approach: we estimate learning-to-rank models M_i for each individual training query Q_i and ask "which of the M_i would have predicted the partial annotations the best?" We take this as a similarity measure between the test query Q_t and each training query Q_i. Coming back to the running example, we expect models of history queries to better predict the partial annotations if the test query is about history as well. Following this intuition, we derive a ranking for the test query as a combination of the most similar learning-to-rank models M_i.

Our approach is related in spirit to Malisiewicz et al. [12]. They create a number of object detectors each of which consists of a positive example and a number of negative ones. They conclude that their approach is better than creating a detector using all of the positive examples together. In another work, McCallum et al. [13] point out that the reduction in the computational cost obtained by dividing the data into overlapping subsets—called canopies—in the context of efficient clustering can be performed without any performance loss.

The main contributions of this paper are as follows: We focus on an approach in which a retrieval model is customized for a given test query by considering the similarity of the test query to the queries available in the training set in the context of learning-to-rank. Our aim here is to improve retrieval results for a given test query by exploiting partial information from an initial search for learning-to-rank problems. We show that our approach provides statistically significant improvements over two baselines; a learning-to-rank (support vector machine (SVM)-rank) baseline with no feedback and a learning-to-rank (SVM-rank) baseline with conventional relevance feedback.

2. RELATED WORK

Here we briefly summarize the related work about modifying learning-to-rank models to improve retrieval. Lv and Zhai [11] point out that the balance parameter of feedback to the original query is static i.e., not adaptive for each query. They propose the idea of adapting the balance variable for each query to increase the retrieval accuracy. When finding the optimal balance parameter for a query and a feedback model, they focus on the following heuristics: 1) discrimination of a query, 2) discrimination of feedback documents, and 3) divergence between a query and feedback documents. Cao et al. [3] indicate that in the training set, the number of relevant documents may vary by query so that the final model moves toward the queries having more relevant documents. They address this issue by adding another parameter to the objective function of the Support Vector Machines (SVM) that balances the effect of the queries on the final model so that they are less biased to the queries having more relevant documents. Zhang et al. [19] focus on a semi-supervised approach to capture the query-specific features such as the unique expansion terms based on the target query in the context of real-time Twitter search. Yue and Joachims [18] focus on finding a retrieval function that is close to the optimal one by formulating it as a bandit algorithm. Hofmann et al. [6] study an online learning-to-rank algorithm that works with implicit feedback as well as balancing exploration and exploitation. Rather than modifying the objective function in the ranker, our approach develops a better retrieval model using queries from the training set that are similar to the test query.

There has also been work on selecting a portion of the training queries most similar to a test query to create better models in the absence of feedback. Geng et al. [4] essentially find the k-nearest neighbors for the test query and create a new model from these k-nearest neighbors. They also discuss modifications to speedup this process. Publicly available datasets such as OHSUMED [5, 9] only have a small number of training queries. On that dataset, we empirically show that this approach does no better than a baseline SVM-rank algorithm. Geng and colleagues in fact specifically use a private dataset with a large number of queries and mention that it does not do well on datasets such as Letor [16] with a small number of queries. Our technique, on the other hand, uses feedback documents to consider a subset of the training set. Peng et al. [15] and Banarjee at al. [1] use a principled component analysis (PCA) based approach to find the nearest training queries. Our approach is to tailor the retrieval model by selecting a number of queries to create a model. There have been some attempts to do this [1, 4]. The motivation in this case was to find queries of a similar type (e.g., navigational or information queries), but no improvements were observed with smaller training sets such as Letor.

3. APPROACH

In this paper, our main focus is to improve the retrieval performance for a given test query exploiting explicit relevance feedback information for that particular query. In addition to using feedback documents to modify the retrieval model, we also use the identified feedback documents to decide which of the training queries are most similar to the test query and consider those for retrieval. To decide which queries are most similar, we first create individual models $M_1, ..., M_k$ using the queries $Q_1, ..., Q_k$ in the training set (e.g., one model per query in the training set). Each of these queries is tested against the feedback documents as measured in normalized discounted cumulative gain (NDCG). A model M_i that achieves a higher NDCG score on the partial judgments for the test query is likely to also provide a better ranking for the remaining documents. We assume that the query Q_i that was used to train M_i with high NDCG is more similar to the test query than queries of models with low NDCG.

Having identified the most similar queries for a test query, we use these queries and the feedback documents to develop a better retrieval model. We make use of the prediction scores obtained from the individual models (a test query Q_t is run against the individual models M_i) and partial model M' is created using the feedback documents. We calculate the final scores to rank documents with Equation 1. S_{final} is the final score of a document in the test query for ranking. Q_S is a subset of the training set containing the queries that are most similar to that particular test query. S_{Q_i} is the prediction score of a document in the test query against the individual model M_i of query Q_i in the training set. $|Q_S|$ is the number of queries in the Q_S set where S' is the score obtained from a partial model M' using the feedback documents.

$$S_{\text{final}} = \frac{1}{|Q_S| + 1}((\sum_{Q_i \in Q_S} S_{Q_i}) + S') \quad (1)$$

4. EXPERIMENTAL DESIGN

Here we provide information about the datasets used in this study, how to perform feature normalization, and the evaluation technique.

Datasets: We focus on the Gov2 dataset that has 25 million documents and 150 queries [14]. In Gov2 there are three relevance ratings, which are 0, *not relevant*; 1, *partially relevant*; and 2, *relevant*. In order to represent document-query pairs, we first retrieve the documents for a given query using the query likelihood model. Then, we focus on the top ranked 1,000 documents for that particular query and extract the features (e.g., low-level content features, high-level content features, and document quality features) defined by Bendersky et al. [2] for each document. The total number of features extracted is 102 and these features are a superset of the features defined by Liu [10]. In order to have a fair comparison among the methods, we only focus on the queries with more than zero and less than ten non-relevant documents in the feedback documents.

We also remove queries without any relevant documents in the test set. There are approximately 42 queries left in each fold after removing such queries. We also analyze the results of our approach on a different dataset; OHSUMED [5] which is also available in Letor [16]. There are 106 queries in the collection split into five folds.

Feature Normalization: Absolute values of a feature for different queries might be in different ranges. For this reason, we perform query-based normalization for each feature [9]. Each feature (x_i) in each document is normalized considering other documents in the same query: $\frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$ where ($min(x_i)$) and ($max(x_j)$) are the minimum and maximum values respectively of x_i for all documents in the same query.

Evaluation and Learning: We split the dataset into three folds each of which consists of a training, a validation, and a test set. We use triple fold cross validation so that each fold becomes a training, a validation, and a test set once. For each fold, we use the validation set to tune the parameters of the ranker. We focus on SVM-rank [7, 8] with linear kernel as a learning-to-rank approach. In order to compute the evaluation scores, we make use of the tools provided in (`research.microsoft.com/~letor/`) [9]. When we report the results for MAP and meanNDCG, we take the average across the folds. The evaluation of our approach and the baselines are based on the documents that are not part of the feedback documents.

5. EXPERIMENTS

In this section, we first explain the baselines that we use to compare our approach. Then we detail the experimental results and discuss the efficiency of our approach.

5.1 Baselines

In order to compare the effectiveness of our approach, we focus on two baselines. The first one considers a learning-to-rank model with no relevance feedback. We also consider another baseline where we apply standard relevance feedback to learning-to-rank models using partial ground truth in top 10 initial ranking. In order to perform relevance feedback, we combine the initial model M_0 and another model M' using the feedback documents (we assume that we know the relevance judgments of these documents) to improve the retrieval in the context of learning-to-rank. According to our validation experiments, this linear combination provides better performance than methods in which the retrieval model is re-trained after inserting the feedback documents to the training set.

As discussed previously, Gent et al. [4] propose a k-nearest neighbor technique for selecting the most similar queries to the test set without feedback. We also consider their method yet another baseline to compare our approach.

5.2 Results and Discussion

We disregard a number of the most dissimilar queries to the test query and consider a subset from the training set so that it provides a better model while ranking the documents. In the experiments, we identify 5%, 10%, 25%, 50%, and 75% of the most dissimilar queries and consider the rest of the queries in the training set (i.e., 95%, 90%, 75%, 50%, and 25% of the most similar queries) to the test query. Consider an example where the relative similarities of queries Q_1, Q_2, Q_3, and Q_4 to the test query Q_t are as follows; $Q_2 > Q_1 > Q_4 > Q_3$. In other words the NDCG result for M_2 (a model created using only Q_2 in the training set) is larger than the NDCG for M_1. In the "50%" case, Q_4 and Q_3 are the most dissimilar queries. For the "25%" case, the query Q_3 is the most dissimilar query to the test query.

We re-implement the approach of Geng et al. [4] and test on the OHSUMED dataset for different k values. According to the experimental results $k = 20$ provides the best MAP and meanNDCG results for the OHSUMED dataset. It turns out that their approach (MAP 44.11% and a meanNDCG score of 51.09%) is not better than our SVM-rank baseline (MAP 44.75% and meanNDCG 53.45%). (note in this particular case, since there is no feedback we use all documents –including the top-10– to compute the numbers). Given that their approach is not better than our SVM-rank (without relevance feedback) baseline, we only provide the SVM-rank baseline without relevance feedback and with relevance feedback in comparison to our approach.

Table 1 shows MAP and meanNDCG scores for the baselines as well as the scores of our approach. First two rows (SVM-rank w/o RF and SVM-rank w/ RF) provide the results for the baselines on the Gov2 dataset and the rest shows the results of our approach in the same dataset. We ignore 5%, 10%, 25%, 50%, and 75% of the most dissimilar queries from the training set. We report retrieval effectiveness in terms of MAP and NDCG in Table 1.

Table 1: Gov2 results on our approach (5%, 10%, 25%, 50%, and 75%) as well as the baselines (SVM-rank w/o RF and SVM-rank w/ RF). Method: 5%, 10%, 25%, 50%, and 75% of the most dissimilar queries are ignored from the training set based on the partial information out of the test query.

Method	MAP	meanNDCG
SVM-rank w/o RF	0.3761	0.5875
SVM-rank w/ RF	0.3825	0.5965
5%	0.3910	0.5981
10%	0.3933	0.6005
25%	0.3984	0.6068
50%	**0.4006**	**0.6116**
75%	0.3960	0.6073

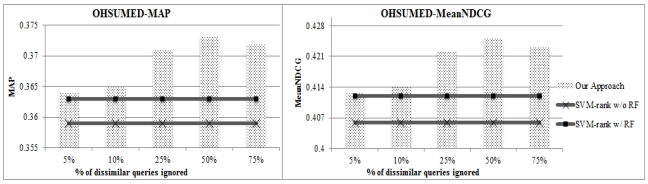

Figure 2: MAP and meanNDCG scores for the OHSUMED dataset.

We obtain the best results when we ignore half of the most dissimilar queries from the training set (see Table 1). The results for the 25% case are very close to the 50% case. When more queries are ignored, some number of similar queries are most likely ignored as well. Likewise, when we ignore fewer queries, then we will leave some number of dissimilar queries in the training set. Our best scores (obtained for the 50% case) provide statistically significant improvements ($\rho < 0.05$) over the baseline without relevance feedback (SVM-rank w/o RF in Table) in all of the folds for MAP and meanNDCG, and for majority of the folds for MAP and meanNDCG when we consider the baseline with relevance feedback (SVM-rank w/ RF in Table). When we compare two baselines (i.e., a learning-to-rank (SVM-rank) baseline with no feedback, and a learning-to-rank baseline with relevance feedback), relevance feedback shows improvements over the case where no feedback is applied.

Analyzing the results on the OHSUMED [5] dataset; in Figure 2 with respect to MAP and meanNDCG, we also observe a very similar pattern to the Gov2 set. Ignoring 50% of the dissimilar queries from the training set yields the best results in terms of MAP and meanNDCG. We evaluate our approach for the case where different numbers of documents are available for training and test queries. The number of documents sub-sampled to create learning-to-rank representations are much lower in the OHSUMED case than the Gov2 case (1000 vs. approx. 200).

In our approach, we create individual models for the queries in the training set. However, this process can be completed offline since the training set is available ahead of time and the queries in the training set are independent of the test query. The operations such as creation of a model using the feedback documents and testing feedback documents against individual models should be performed online. However, these processes can be completed in a couple of milliseconds since we only use a small number of feedback documents (10 in our experiments).

6. CONCLUSION

In this work we explore a query-specific modification of learning-to-rank approaches. We make use of the feedback documents to improve a retrieval model using queries in the training set that are similar to a particular test query. We evaluate our approach on the Gov2 and OHSUMED dataset with two baselines; a learning-to-rank (SVM-rank) baseline with no relevance feedback and a learning-to-rank (SVM-rank) baseline with standard relevance feedback using partial ground truth. In our experimental evaluation, we obtain statistically significant improvements using our approach over such baselines by exploiting the partial information of a particular test query.

Several future directions are promising. First, we would like to study the effect of the number of available feedback judgments. Next, we target query-based similarity functions to avoid the need for feedback judgments.

7. ACKNOWLEDGEMENTS

This work was supported in the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. I would like to thank Laura Dietz for her valuable comments.

8. REFERENCES

[1] S. Banarjee, A. Dubey, J. Machchhbar, and S. Chakrabarti. Efficient and accurate local learning for ranking. In *Learning-to-rank Worshop SIGIR*, 2009.

[2] M. Bendersky, W. B. Croft, and Y. Diao. Quality biased ranking of web documents. In *WSDM*, 2011.

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

[4] X. Geng, T. Liu, T. Qin, A. Arnold, H. Li, and S. H. Query dependent ranking using k-nearest neighbor. In *SIGIR*, 2008.

[5] W. Hersh, C. Bukley, T. Leone, and D. Hickman. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR*, 1994.

[6] K. Hoffmann, S. Whiteson, and M. Rijke. Balancing exploration and exploitation in learning to rank online. In *ECIR*, pages 251–263, 2011.

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

[8] T. Joachims. Training linear SVMs in linear time. In *KDD*, 2006.

[9] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmarking learning to rank for information retrieval. In *SIGIR*, 2007.

[10] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in IR*, 3(3):225–331, 2009.

[11] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *CIKM*, pages 255–264, 2009.

[12] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of Exemplar-Svms for object detection and beyond. In *ICCV*, 2011.

[13] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.

[14] D. Metzler, T. Strohman, H. Turtle, and W. B. Croft. Indri at trec 2004: Terabyte track. Technical report, DTIC Document, 2004.

[15] J. Peng, C. Macdonald, and I. Ounis. Learning to select a ranking function. In *ECIR*, pages 114–126, 2010.

[16] T. Qin, T.-Y. Liu, J. Xu, and H. Li. Letor: A benchmark collection for research on learning to rank for IR. *Information Retrieval*, 13(4):346–374, 2010.

[17] A. Trotman. Learning to rank. *Information Retrieval*, 8(3):359–381, 2005.

[18] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, 2009.

[19] X. Zhang, B. He, T. Luo, and B. Li. Query-biased learning to rank for real-time twitter search. In *CIKM*, 2012.