Document Selection Methodologies for Efficient and Effective Learning-to-Rank

Javed A. Aslam†, Evangelos Kanoulas†, Virgil Pavlu†, Stefan Savev†, Emine Yilmaz*

†College of Computer and Information Science
Northeastern University
360 Huntington Ave, #202 WVH
Boston, MA 02115
{jaa,ekanou,vip,savev}@ccs.neu.edu

*Microsoft Research Cambridge
7 JJ Thomson Avenue
Cambridge CB3 0FB, UK
eminey@microsoft.com

ABSTRACT

Learning-to-rank has attracted great attention in the IR community. Much thought and research has been placed on query-document feature extraction and development of sophisticated learning-to-rank algorithms. However, relatively little research has been conducted on selecting documents for learning-to-rank data sets nor on the effect of these choices on the efficiency and effectiveness of learning-to-rank algorithms.

In this paper, we employ a number of document selection methodologies, widely used in the context of evaluation – depth-k pooling, sampling (infAP, statAP), active-learning (MTC), and on-line heuristics (hedge). Certain methodologies, e.g. sampling and active-learning, have been shown to lead to efficient and effective evaluation. We investigate whether they can also enable efficient and effective learning-to-rank. We compare them with the document selection methodology used to create the LETOR datasets.

Further, all of the utilized methodologies are different in nature, and thus they construct training data sets with different properties, such as the proportion of relevant documents in the data or the similarity among them. We study how such properties affect the efficiency, effectiveness, and robustness of learning-to-rank collections.

Categories and Subject Descriptors: H. Information Systems; H.3 Information Storage and Retrieval; H.3.3 Information Search and Retrieval:Retrieval models

General Terms: Experimentation, Measurement, Theory

Keywords: Learning-to-Rank, Document Selection Methodologies, Sampling, Evaluation

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* 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 feature extraction and the development of sophisticated learning-to-rank algorithms. However, relatively little research has been conducted on the choice of queries and documents for learning-to-rank data sets nor on the effect of these choices on the ability of a learning-to-rank algorithm to "learn", effectively and efficiently.

Constructing data sets for learning-to-rank tasks requires assembling a document corpus, selecting user information requests (queries), extracting features from query-document pairs and annotating documents in terms of their relevance to these queries (annotations are used as labels for training). Over the past decades, document corpora have been increasing in size from thousands of documents in the early TREC collections to billions of documents/pages in the World Wide Web. Due to the large size of document corpora it is practically infeasible (1) to extract features from all document-query pairs, (2) to judge each document as relevant or irrelevant to each query, and (3) to train learning-to-rank algorithms over such a vast data set.

The main bottleneck in constructing learning-to-rank collections is annotating documents with relevance grades. It is essential therefore, both for the efficiency of the construction methodology and for the efficiency of the training algorithm, that only a small subset of documents be selected. The document selection, though, should be done in a way that does not harm the effectiveness of learning.

LETOR [14] is the only attempt made to construct a publicly available learning-to-rank collection. Documents, queries and relevance judgments were obtained from the OHSUMED and TREC test collections. Since there are many documents in these collections, in order to reduce the computational effort required to extract features and train ranking functions over these data sets, only a subset of them was chosen in the following way: Documents were first ranked by their BM25 [12] score, which is known to correlate well with the relevance of a document to a query.

SIGIR'09, July 19–23, 2009, Boston, Massachusetts, USA.

Features then were extracted only from the corresponding top 1000 documents, in an effort to include as many relevant documents as possible in the learning-to-rank dataset. Features were also extracted from documents that were not ranked in this top 1000 but were judged as relevant in the corresponding TREC collections.

Even though LETOR has been widely used by many researchers, recent work demonstrated bias in this document selection methodology that could harm learning-to-rank algorithms [15, 17], hence the collection has been criticized. When the LETOR collection was built, the fact that documents with low BM25 score were selected only if they were relevant resulted in BM25 being negatively correlated with relevance in the LETOR collection. This is a highly counter-intuitive outcome. To avoid the aforementioned implication, these extra documents with low BM25 scores were dropped in the latest LETOR release [13].

For the OHSUMED learning-to-rank collection only judged documents were selected for feature extraction. As pointed out by Minka and Robertson [15], this selection methodology results in an atypical proportion of relevant and non-relevant documents in the collection. Further, the nature of the non-relevant documents in the learning-to-rank collection is not representative of that in the entire OHSUMED collection.

These issues clearly manifest the effect a document selection methodology may have on the effectiveness of the learning-to-rank algorithms, and thus, on the performance of the resulting retrieval systems. Furthermore, the conclusions about the relative quality of different learning-to-rank algorithms may not be reliable.

Unlike document selection for learning-to-rank, where little work has been done, a significant volume of work has appeared in the literature regarding document selection for efficient and effective evaluation of retrieval systems [2, 5, 6, 21]. Most of these methods are based on sampling documents in an intelligent way to minimize judgment effort and/or using statistical techniques to compute the values of traditional evaluation measures efficiently.

In this work, we explore the duality between document selection methodologies for evaluation and document selection methodologies for learning-to-rank. The main question we ask is : "Can any of the methodologies designed for efficient evaluation also be used for constructing effective learning collections? If yes, which one of these methods is better for this purpose?"

We employ five different document selection methodologies that are well studied in the context of evaluation, along with the method used in LETOR for comparison purposes. Subsets of documents are chosen according to the six methods at different percentages of the complete document collection (in our case the depth-100 pool), and features are extracted from the selected query-document pairs. State of the art learning-to-rank algorithms are used then to train ranking functions over each one of the data sets the six selection methods have produced, and the resulting functions are compared with each other in terms of their performance.

In particular, (1) we explore whether certain document selection methodologies are better than others in terms of both efficiency and effectiveness; that is, how fast, in terms of documents, can ranking functions learn to combine features in a meaningful way over the corresponding data sets such that their performance is not significantly worse than the performance of functions trained over the complete collection (depth-100 pool), and (2) we isolate certain characteristics of the selected subsets of documents (e.g. the percentage of relevant documents, or the similarity among relevant documents in the subsets) and study their effect on the efficiency and effectiveness of learning.

2. METHODOLOGY

In order to investigate the effect of document selection on the ability of learning-to-rank algorithms to effectively and efficiently learn a ranking function, five different document selection methodologies, widely used in retrieval evaluation, are studied.

Our *complete* document collection consists of the depth-100 pools from TREC 6, 7 and 8 adhoc tracks [1]. This collection consists of 150 queries in total, along with the corresponding relevance judgments. Features are extracted from all query-document pairs. Using different document selection methodologies, for each query, documents from the complete collection are selected with different percentages from 0.6% to 60%, forming different sized subsets of the complete collection for each methodology.

Features and relevance judgments pairs are then partitioned into five parts in order to conduct five-fold cross validation. For each fold, three parts are used for training, one part for validation and one part for testing. The documents in the training and validation sets are samples of the complete collection, as described above. The testing set consists of the complete set of documents.

State of the art learning-to-rank algorithms are then trained and the quality of the resulting ranking models is assessed by mean average precision (MAP).

2.1 Data sets

The document corpus, the queries and the relevance judgments are obtained from TREC 6, 7 and 8 ad-hoc retrieval track.

The document corpus consists of approximately half a million documents (528,155) from the Financial Times, the Federal Register, the Foreign Broadcast Information Service and the LA Times document collections [20].

The queries were developed by NIST assessors and they were chosen based on their estimated number of relevant documents in the corpus. Collectively, 150 such queries were developed. A number of retrieval systems were run over these queries and for each query the depth-100 pools of the returned documents were judged as relevant or irrelevant by the same NIST assessor that issued the query.

We extract features from all query-document pairs. The features extracted are shown in Table 2.1 and they are a subset of the LETOR3.0 features [13]. The description of these features along with their exact formulas can be found in the LETOR3.0 documentation [13]. Each feature is computed over the document text (without the title) and over the document text combined with the title, resulting in 22 features in total. Note that web features such as PageRank are not computed since the document corpus is not a web corpus. The language modeling features are implemented according to Zhai and Lafferty [22] while the BM25 feature is implemented according to Singhal [18].

[1]In the sections that follow we use the "depth-100 pools" and "complete collection" interchangeably

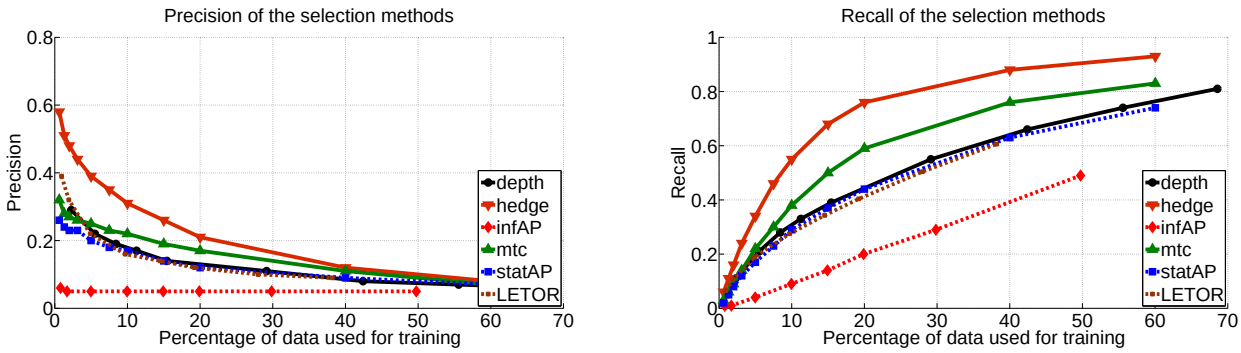

Figure 1: Precision and Recall of the selected documents for different selection strategies and for different sampling percentages.

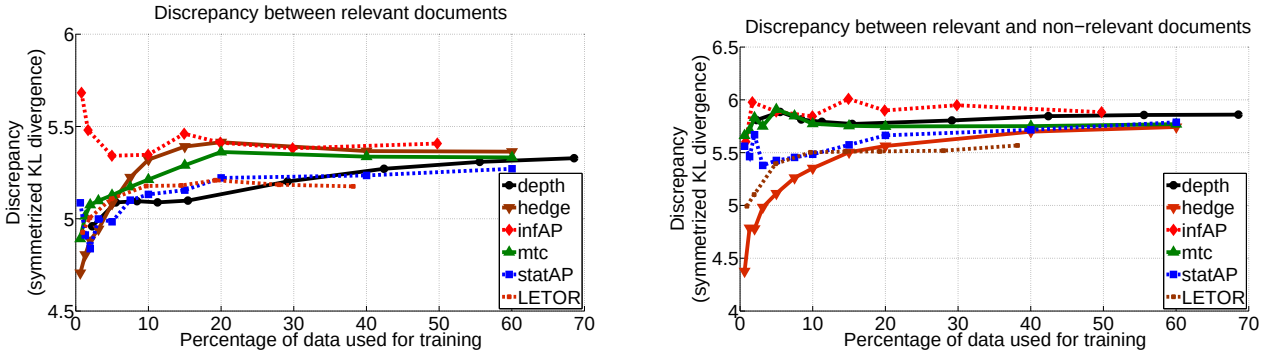

Figure 2: Discrepancy among relevant documents and among relevant and non-relevant documents selected by different selection strategies and for different sampling percentages.

	Features
1.	BM25
2.	LogBM25_Feature
3.	LM_ABS_Feature
4.	LM_DIR_Feature
5.	LM_JM_Feature
6.	LogNormalizedTF_Feature
7.	SumLogTF_Feature
8.	TF_Feature
9.	TF_IDF_Feature
10.	LogTF_IDF_V2_Feature
11.	NormalizedTF_Feature

Table 1: Feature Set

2.2 Document selection

For each query, six different document selection methodologies are employed to choose documents from the complete collection:

- **Depth-k pooling:** According to the *depth pooling*, the union of the top k documents retrieved by each retrieval system submitted to TREC in response to a query is formed and only the documents in this depth-k pool are selected to form the learning-to-rank collection. The intuition behind depth-pooling is that most relevant documents appear at the top of the ranked list and therefore depth-k pools contain most of them [9, 23].

- **InfAP sampling (uniform random sampling):** InfAP sampling [21] utilizes uniform random sampling to select documents to be judged. In this manner, the selected documents are representative of the documents in the complete collection.

- **StatAP sampling (stratified random sampling):** StatAP sampling [16] employs stratified sampling. Using a prior of relevance induced by the average precision measure, each document is selected with probability roughly proportional to its likelihood of relevance.

- **MTC (greedy on-line algorithm):** MTC [5] is a greedy on-line algorithm that selects documents according to how informative they are in determining whether there is a performance difference between two retrieval systems. Our intuition is that many of the documents selected by MTC, relevant or not, are "interesting" for learning by being relatively close to the decision surface of the classifier, similar to active learning.

- **Hedge (greedy on-line algorithm):** Finally, hedge is an on-line learning algorithm used to combine expert advice. It is essentially a feedback-metasearch

technique, which, when applied to the document selection problem, aims at choosing documents that are most likely to be relevant [1]. Hedge finds many relevant documents "common" to various retrieval systems, thus documents likely to contain many of the query words.

- **LETOR:** For comparison purposes, a LETOR-like document selection methodology is also employed. According to this methodology, documents in the complete collection are first ranked by their BM25 scores for each query and the top-k documents are then selected for feature extraction. This method is designed to select documents that are considered relevant to the query by BM25.

When the properties of the above document selection methodologies are considered, one can see that infAP creates a representative selection of documents, statAP and depth-k pooling aim at identifying more relevant documents utilizing the knowledge that retrieval systems return relevant documents at higher ranks, the LETOR-like method aims at selecting as many relevant documents according to BM25 as possible, hedge aims at selecting only relevant documents, and MTC greedily selects discriminative documents.

For all these strategies, the precision, computed as the ratio of the number of relevant documents in the document sample to the total number of documents in the document sample, and the recall, computed as the ratio of relevant documents in the document sample to the total number of relevant documents in the complete collection, are illustrated in the left and right plots of Figure 1, respectively. As expected, hedge achieves the largest values of precision and recall, followed by MTC. Pooling and statAP follow by achieving similar values of precision and recall. Since infAP is based on uniform random sampling, the precision of infAP stays constant while the recall grows linearly with the sample percentage. The LETOR-like selection achieves both high precision and recall at small percentages of data used for training (up to 5%) and then it drops to the levels of statAP and depth pooling.

Further, the discrepancy among the selected relevant documents, along with the discrepancy among the selected relevant and non-relevant documents for the different selection methods is illustrated in Figures 2. The discrepancy is measured for each pair of documents by the symmetrized Kullback-Leibler divergence between the documents' (smoothed) language models, then averaged over all pairs in a set. As it can be observed at the leftmost plot, for all methods except infAP, the selected documents are very similar to each other. For small percentages, it can be seen that the relevant documents picked by hedge are very similar to each other. As more documents are selected according this algorithm, relevant documents with different properties can be identified. Depth-pooling and statAP select similar relevant documents due to the underlying retrieval systems that return similar relevant documents at the top-end of their ranked lists, while hedge picks similar relevant documents by design. In particular, hedge selects very similar documents regardless of their relevance, as it can be observed in the right-most plot. At the other end of the discrepancy scale, infAP for small sampling percentages selects the most diverse relevant documents while it converges fast to the average discrepancy between documents in the complete collection. The LETOR-like selection methodology also selects very similar documents, since the documents selected are those that give high BM25 values and thus have similar characteristics.

2.3 Learning-to-rank algorithms

We employ five different learning-to-rank algorithms to test the document selection methodologies,

- **RankBoost (boosting):** RankBoost is a very well known ranking mechanism based on the AdaBoost algorithm [8] for supervised learning. RankBoost training is performed using pairwise preferences, essentially combining several "weak" rankers into a master one using on-line learning. Typical weak learners are features of the data (in our case extracted features from documents) with a threshold that best differentiates the relevant documents from non-relevant ones; however, in general, the weak rankers can be very complicated retrieval/ranking functions.

  Rankboost is widely reported in many learning-to-rank publications [13], primarily as a baseline ranking algorithm. In our experiments, we run the algorithm for 50 epochs. In some tests we trained Rankboost for larger number of epochs (up to 1000) and concluded that performance after 50 epochs was stable, even for small datasets.

- **Regression (regression):** We implemented a baseline linear regression ranker, with the purpose of studying the change in learning performance across various training sets. The procedure basically fits a linear regression model to the training set, and then it uses the learned model to predict (and rank) the test documents.

- **Ranking SVM (SVM):** The implementation of Support Vector Machines used is based on SVM-perf [10, 11] and is similar to the ones reported in ranking literature [13]. We use a polynomial kernel of degree 2 (sparse approximation with 400 basic functions), and a tradeoff constant of $c = 0.01$. We experimented with several loss functions; the results presented here use as loss function a variant of ROC area, specifically the percentage of swapped positive/negative pairs. The structural learning algorithm uses a shrinking heuristic[2] in order to speed up the training.

- **RankNet (neural network):** RankNet [3] is one of the basic learning-to-rank algorithms based on neural networks. The algorithm is based on training the neural net on pairs of documents (or feature vectors) that have different relevance. During training, single feature vectors are first forward propagated through the net and are sorted based on their scores. The RankNet cost is a sigmoid function followed by a cross entropy cost that evaluates the difference of the *learned* probability that a document pair will be ranked in some order from the *actual* probability. In our experiments, the training was run for 300 epochs (no significant improvement was observed if more epochs were used).

- **LambdaRank (neural network):** LambdaRank [4] aims at directly optimizing an IR metric, in particular,

[2]http://svmlight.joachims.org/svm_perf.html

NDCG. Since the IR metrics are not smooth as they depends on ranks of documents, it uses the approach of defining the gradient of the target evaluation metric only at the points needed. Given a pair of documents, the gradients used in LambdaRank are obtained by scaling the RankNet cost with the amount of change in the value of the metric obtained by swapping the two documents. Similar to RankNet, LambdaRank training was also run for 300 epochs.

As a summary, RankBoost optimizes for pairwise preferences, Regression optimizes for classification error in the relevance judgments, and SVM optimizes for the area under the ROC curve. RankNet aims to optimize for the probability that two documents are ranked in correct order in the ranking. Finally, LambdaRank directly optimizes for nDCG and even though the gradients are virtually defined, the method is shown to find the local optimum for the target metric.

All the algorithms, with the exception of Regression, are "pair-wise" because they consider pairs of documents while training, either directly in the learning mechanism or indirectly in the loss function.

3. RESULTS

The performance of the learning-to-rank algorithms when trained over the different data sets produced by the six document selection methodologies is illustrated in Figure 3. The x-axis on the plots is the percentage of documents sampled from the complete document collection (depth-100 pool). The performance of the rankers (y-axis) is measured by the mean average precision (MAP) of the ranked list of documents returned by the rankers in response to the queries in the testing data sets. Each one of the document selection methods employed corresponds to a curve in the plot.

As one can observe in Figure 3, for most of the cases, the learning-to-rank algorithms reach almost optimal performance with as little training data as 1% of the complete collection. The Student's t statistical test was employed in order to test whether the difference among the achieved MAP scores of the ranking function for different sampling percentages and the maximum MAP score the ranking functions obtain (MAP using full training data) are statistically significant. The t-test exhibits no significant differences for ranking functions trained over infAP, statAP, depth-pooling and MTC at any of document sampling percentage above 2%.

Therefore, training data sets whose sizes are as small as 1% to 2% of the complete collection are just as effective for learning-to-rank purposes as the complete collection. Thus, one can train much more efficiently over a smaller (though effectively equivalent) data set or at equal cost one can train over a far "larger" and more representative data set either by increasing the number of queries of by selecting documents deeper in the rankings. Note that the number of features used by the learning-to-rank algorithms may well affect the efficiency of these algorithms to learn an effective ranking function [19]. In our experiments only twenty two (22) features where used, most of which are ranking functions of their own (e.g. BM25 or language models). Therefore, in the case where hundreds of (raw) features are employed, ranking functions may need more than 1% of the complete collection to achieve optimal performance. Nevertheless, in a setup similar to LETOR setup, as in our experiments, we show that substantially less documents than the ones used in LETOR can lead to similar performance of the trained ranking functions.

Furthermore, it is apparent from Figure 3 that the effectiveness of small data sets for learning-to-rank purposes eminently depends on the document selection methodology employed. The most striking example of an inefficient document selection methodology is that of hedge. Ranking functions trained on data sets constructed according to the hedge methodology only reach their optimal performance when trained over data sets that are at least 20% of the complete collection, while in the worst case, the performance of some ranking functions is significantly lower than the optimal one even when trained over 40% to 50% of the complete collection (e.g. the performance of RankBoost, Regression and RankNet with the hidden layer).

Ranking functions exhibit their second worst performance when trained over data sets constructed according to the LETOR-like document selection methodology. Even though LETOR data sets have been widely used by researchers, our results show that the document selection methodology employed in LETOR is neither the most effective nor the most efficient way to construct learning-to-rank collections.

The deficiency of learning-to-rank data sets produced according to hedge and LETOR-like document selection methodologies may seem counterintuitive. One would expect relevant documents to be much more "informative" than non-relevant documents for the purpose of learning-to-rank, and both hedge and LETOR-like document selection methodologies are designed to choose as many relevant documents as possible. Clearly this is not the case according to our results.

In what follows, we try to give an explanation of these counterintuitive. Figure 4 illustrates the performance of two of the learning-to-rank algorithms (SVM and RankBoost) for which data sets created according to hedge and LETOR-like methods seem the least effective. The y-axis corresponds to the performance of the ranking functions. The x-axis in the top-row plots corresponds to the percentage of relevant documents in the learning-to-rank data sets, while at the bottom-row plots, it corresponds to the discrepancy among the selected relevant and non-relevant documents. Each dot in these plots corresponds to a training data set at some sampling percentage, regardless of the document selection algorithm employed. As can be observed, there is a clear negative correlation between the percentage of relevant documents (above a certain threshold) and the performance of both ranking functions. Further, a strong positive correlation is exhibited between the dissimilarity among relevant and non-relevant documents and the performance of the two algorithms. This is a strong indication that over-representation of relevant documents in the training data sets may harm the learning-to-rank algorithms. Furthermore, when relevant and non-relevant documents in the training data set are very similar to each other, performance of the resulting ranking functions decline.

Both hedge and LETOR-like document selection methodology, by design, select as many relevant documents as possible. As shown in Figure 2, the documents selected by the two methods also exhibit very high similarity to each other.

In contrast to the aforementioned selection methodologies that are designed to select as many relevant documents as possible, infAP, statAP, depth-pooling and MTC tend to

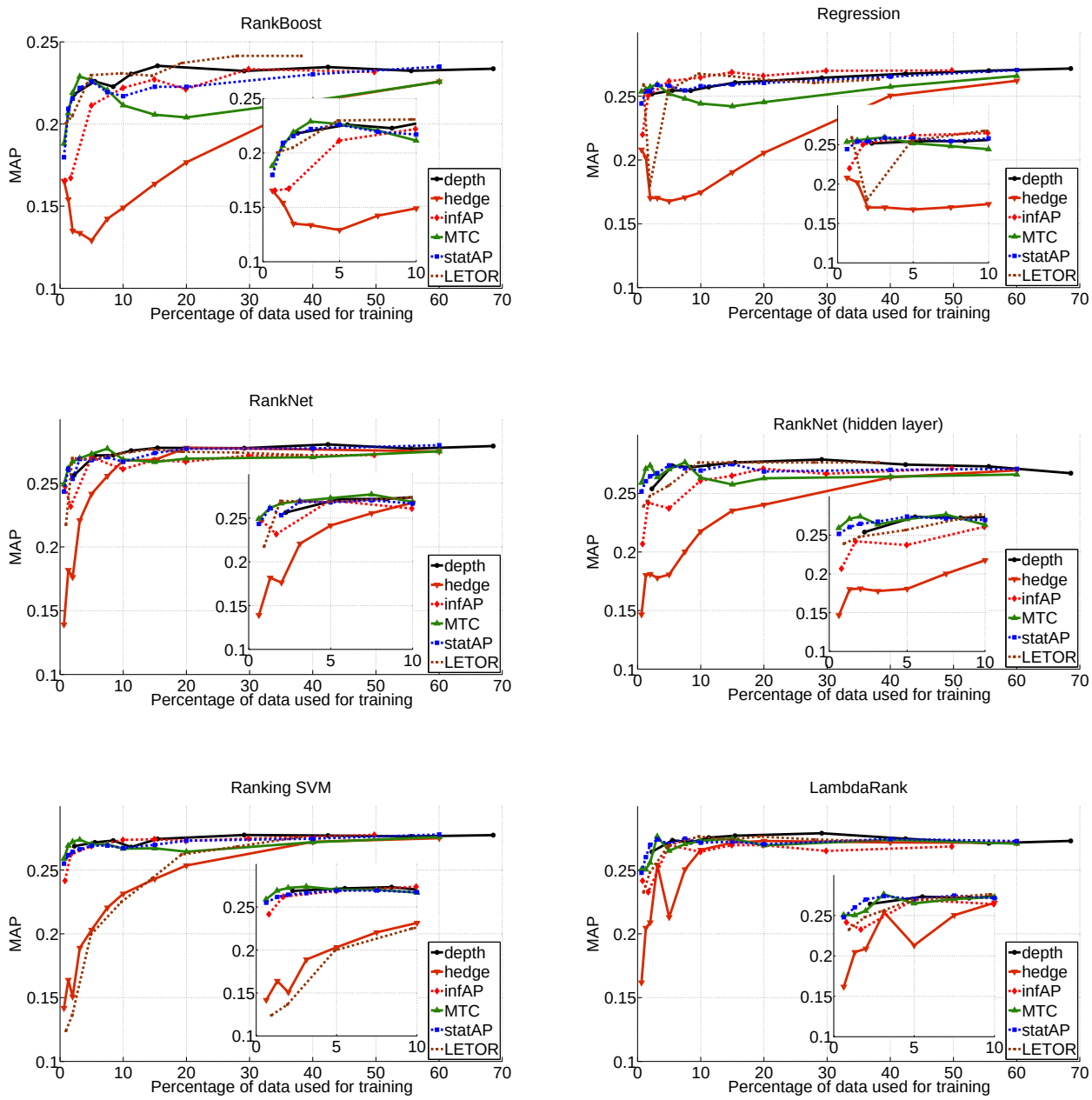

Figure 3: RankBoost, Regression, Ranking SVM, RankNet (linear and nonlinear) and LambdaRank ranking performance across several document selection methods at various sampling percentages.

construct data sets that are more representative of the complete collection. In particular, infAP randomly samples the complete collection, and thus, all documents in the collection have equal probability to be included in the training data set, regardless of their relevance grade. Even though depth-pooling tends to select documents from the top end of the ranked lists of the underlying participating systems, it treats all systems in a fair manner regardless of their quality, giving poor underlying systems an equal opportunity to contribute to the constructed data sets. Thus, many non-relevant documents are included in the resulting training sets. StatAP selects the documents that are commonly retrieved by multiple systems due to the way the sampling prior is computed. Hence, the quality of the sampled documents depend on the quality of the underlying systems. If a non-relevant document is retrieved by many retrieval systems, then this document is still very likely to be sampled by the statAP sampling. Finally, MTC, by design, selects the most informative documents for the purpose of evaluation, regardless of their relevance.

Therefore, the percentage of relevant documents in the learning-to-rank data sets along with the similarity of relevant and non-relevant documents appear to be good explanatory parameters for the efficiency and effectiveness of all the aforementioned document selection methodologies.

In order to quantify the explanatory power of these two

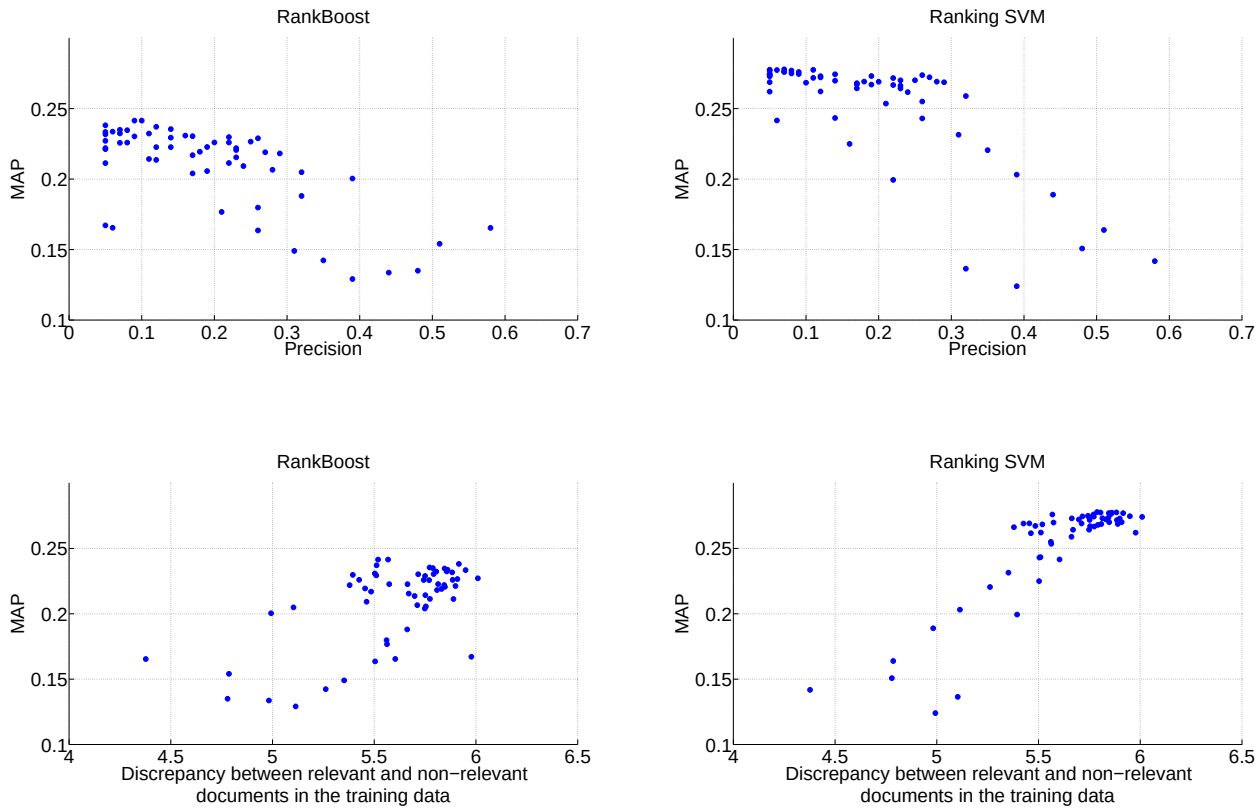

Figure 4: Scatter plots of the performance of RankBoost and SVM ranking functions versus the percentage of relevant documents and the discrepancy between relevant and non-relevant documents in the training data.

parameters, we formulate a linear model, with the performance of RankBoost and SVM as measured by MAP over the testing data sets expressed as a linear function of the percentage of relevant documents and the dissimilarity among relevant and non-relevant documents in the data sets. Both the adjusted R^2 and the F-statistic of the resulting linear models indicate an excellent goodness of fit, with the former being equal to 0.99 and with the p-value of the latter being equal 10^{-16}.

Further, to assess the relative importance of the two explanatory parameters, with respect to the performance of the learning-to-rank algorithms, we also fit an ANOVA model to the data and performed a variance decomposition analysis, according to which the percentage of relevant documents accounts for about 60% of the variance in the MAP scores across all data sets and the discrepancy between relevant and non-relevant documents accounts for about 39%. Therefore, we conclude that both the proportion of relevant documents and the dissimilarity between documents are highly important for the quality of a learning-to-rank collection, with the former affecting the quality more than the latter.[3]

Finally, by revisiting Figure 3, one can observe that some learning-to-rank algorithms are more robust to document selection methodologies than others. In particular, LambdaRank and RankNet seem to be more robust than Regression, RankBoost and Ranking SVM. To assess the relative importance between the learning-to-rank algorithms' effect on MAP and the selection methodologies' effect on MAP, we fit a 2-way ANOVA model to the MAP values and again perform a variance decomposition. According to ANOVA 26% of the variance in the MAP scores is due to the selection methodology and 31% due to the learning-to-rank algorithm, while 5% is due to the algorithm-selection methodology interaction. When we perform the same analysis to MAP values over datasets up to 10% of the complete collection, then 44% of the variance in the MAP scores is due to the selection methodology and 23% is due to the learning-to-rank algorithm, while 10% is due to the algorithm-selection methodology interaction.

4. CONCLUSIONS

In this paper, we analyzed the problem of building document collections for efficient and effective learning-to-rank. In particular, we explored (1) whether the algorithms that are good for reducing the judgment effort for efficient evaluation are also good for reducing judgment effort for efficient learning-to-rank and (2) what makes one learning-to-rank collection better than another.

For this purpose we constructed different sized learning collections employing depth pooling, infAP, statAP, MTC, hedge and the LETOR methodology. We then ran a number of state of the art learning-to-rank algorithms over these

[3]Note that we have tested the effect of other explanatory parameters to the variability of MAP values (e.g. recall, total number of documents, total number of relevant documents, similarity between relevant documents, interactions between these parameters). None of these parameters appeared to be significant.

training collection and compared the quality of the different methods used to create the collections based on the relative performance of the learning algorithms.

We showed that some of these methods (infAP, statAP and depth pooling) are better than others (hedge and the LETOR method) for building efficient and effective learning-to-rank collections. This is a rather surprising result given the wide usage of the LETOR datasets as it suggests that using the same judgment effort, better collections could be created via other methods.

Furthermore, we showed that both (1) the proportion of relevant documents to non-relevant documents and (2) the similarity between relevant and non-relevant documents in the data sets highly affect the quality of the learning-to-rank collections, with the latter having more impact.

Finally, we observed that some learning-to-rank algorithms (RankNet and LambdaRank) are more robust to document selection methodologies than others (Regression, RankBoost and Ranking SVM).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No IIS-0533625 and IIS-0534482. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

5. REFERENCES

[1] J. A. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch and the efficient evaluation of retrieval systems via the hedge algorithm. In J. Callan, G. Cormack, C. Clarke, D. Hawking, and A. Smeaton, editors, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 393–394. ACM Press, July 2003.

[2] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In S. Dumais, E. N. Efthimiadis, D. Hawking, and K. Jarvelin, editors, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 541–548. ACM Press, August 2006.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM.

[4] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. C. Platt, T. Hoffman, B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 193–200. MIT Press, 2006.

[5] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 268–275, 2006.

[6] B. Carterette, V. Pavlu, E. Kanoulas, J. A. Aslam, and J. Allan. Evaluation over thousands of queries. In S.-H. Myaeng, D. W. Oard, F. Sebastiani, T.-S. Chua, and M.-K. Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 651–658. ACM Press, July 2008.

[7] W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors. *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Aug. 1998.

[8] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[9] D. Harman. Overview of the third text REtreival conference (TREC-3). In D. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 1–19. U.S. Government Printing Office, Apr. 1995.

[10] T. Joachims. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

[11] T. Joachims. Training linear SVMs in linear time. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 217–226, 2006.

[12] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[13] T.-Y. Liu, T. Qin, J. Xu, X. Wenying, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval.

[14] T. Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR '07: Proceedings of the Learning to Rank workshop in the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[15] T. Minka and S. Robertson. Selection bias in the letor datasets. In *SIGIR '08: Proceedings of the of the Learning to Rank workshop 31st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2008. ACM.

[16] V. Pavlu. *Large Scale IR Evaluation*. PhD thesis, Northeastern University, College of Computer and Information Science, 2008.

[17] T. Qin, T.-Y. Liu, J. Xu, and H. Li. How to make letor more useful and reliable. In *SIGIR '08: Proceedings of the of the Learning to Rank workshop 31st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2008. ACM.

[18] A. Singhal and G. Inc. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24:2001, 2001.

[19] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 585–593, New York, NY, USA, 2006. ACM.

[20] E. M. Voorhees and D. Harman. Overview of the seventh text retrieval conference (TREC-7). In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 1–24, 1999.

[21] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In P. S. Yu, V. Tsotras, E. Fox, and B. Liu, editors, *Proceedings of the Fifteenth ACM International Conference on Information and Knowledge Management*, pages 102–111. ACM Press, November 2006.

[22] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[23] J. Zobel. How reliable are the results of large-scale retrieval experiments? In Croft et al. [7], pages 307–314.