Learning to Rank System Configurations

Romain Deveaud, Josiane Mothe, Jian-Yun Nie

Learning to Rank System Configurations

Romain Deveaud Josiane Mothe
IRIT UMR5505, Université de Toulouse, France
firstname.lastname@irit.fr

Jian-Yun Nie
DIRO, Université de Montréal, Québec
nie@iro.umontreal.ca

ABSTRACT

Information Retrieval (IR) systems heavily rely on a large number of parameters, such as the retrieval model or various query expansion parameters, whose values greatly influence the overall retrieval effectiveness. However, setting all these parameters individually can often be a tedious task, since they can all affect one another, while also vary for different queries. We propose to tackle this problem by dealing with entire *system configurations* (i.e. a set of parameters representing an IR system) instead of single parameters, and to apply state-of-the-art Learning to Rank techniques to select the most appropriate configuration for a given query. The experiments we conducted on two TREC AdHoc collections show that this approach is feasible and significantly outperforms the traditional way to configure a system, as well as the top performing systems of the TREC tracks. We also show an analysis on the impact of different features on the model's learning capability.

Keywords

Information retrieval; learning to rank; retrieval system parameters

1. INTRODUCTION

The effectiveness of Information Retrieval (IR) systems heavily relies on a large number of parameters, ranging from the choice of stemmer, the smoothing technique to the number of terms that are added to the query when performing automatic query expansion. Over the years, and through the evaluation forums such as TREC, the IR community has produced abundant field knowledge, scattered in the literature, on setting the appropriate values of these parameters, in order to optimise the performance of the retrieval systems. The parameters are usually studied in isolation: one attempts to set the value for only one or a few parameters at a time, without taking into account the influence of the setting of other parameters. This makes it difficult to configure a globally optimised set of parameters, as modern IR systems involve a quite large number of parameters that are mutually dependent. There has been no systemic study trying to set the best values for all the parameters of a system.

In this paper, we treat a complete set of parameters of a system as a *system configuration*. We cast the problem of selecting the most appropriate system configuration as a configuration ranking problem using a learning to rank (L2R) [8] approach. The candidate space is formed of tens of thousands of possible system configurations, each of which sets a specific value for each of the system parameters. L2R models are trained to rank them with respect to a performance measure. This approach has the advantage of taking all the system parameters into account at the same time, thus allowing them to influence each other. Moreover, our approach can make a query-dependent choice of system configuration, i.e. different search strategies could be selected for different types of query.

To our knowledge, this is the first study using L2R to rank system configurations. The most similar work to ours is [2], which aims to predict system performance using regression models. While the predicted system performance could be used to select the best system configuration, our study is broader, and the regression method is a special case of the L2R techniques we examine.

The main contribution of this paper is that we propose a new way based on L2R to set system parameters and show its feasibility and competitive or superior retrieval effectiveness to the state of the art on two TREC collections.

2. LEARNING TO RANK SYSTEM CONFIGURATIONS

Our method is based on L2R approaches for IR [8]. However, instead of ranking documents for a query, we rank system configurations. The problem is formulated below.

We assume that an IR system involves a set $\mathbf{P}$ of parameters. Each parameter $p_i \in \mathbf{P}$ can take a value from its domain D_i. Therefore, we have $\prod_i |D_i|$ possible configurations (without considering the fact that some configurations are impossible). This number could be very large, given the quite large number of system parameters used in modern systems and their possible values. We also assume that we have a set $\mathbf{Q}$ of queries for which we have relevance judg-

ments on a document collection, which can be used to generate training examples for L2R models: for each possible configuration $c_j \in \mathbf{C}$, where $\mathbf{C}$ is the space of all configurations, we generate a measure in the IR performance metric (such as MAP, P@n, etc.) for the pair (q_k, c_j). Our goal is to rank the possible configurations for a new query q such that the best ranked system configuration could lead to the best performance.

A L2R model is based on a set of features defined on (q_k, c_j). Usually, in L2R approaches, the features are related to the query, the document (and sometimes the document collection), as well as the relationships between them. In this work, we define features in a similar way, by computing features related to the query q_k and to the configuration c_j. Our primary goal in this paper is to test the feasibility of L2R approaches for setting system configurations.

We consider a set of common system parameters (see Table 1): 1 parameter for retrieval model and 4 parameters for pseudo-relevance feedback. Notice that these parameters are all for retrieval (not for indexing) and can be set for a query on the fly. For each of the possible configuration[1], we run the Terrier IR system [11] to obtain the corresponding performance measures. This amounts to more than 10,000 different configurations for each training query.

Table 1: Description of the system parameters that we use to build our dataset

Parameter	Description & values[2]
Retrieval model	21 different retrieval models: DirichletLM, JsKLs, BB2, PL2, DFRee, DFI0, XSqrAM, DLH13, HiemstraLM, InL2, DLH, DPH, IFB2, TFIDF, InB2, InexpB2 , DFRBM25 , BM25, LGD, LemurTFIDF, InexpC2.
Expansion model	7 query expansion models: nil, Rocchio, KL, Bo1, Bo2, KLCorrect, Information, KLComplete.
Expansion documents	Number of documents used for query expansion: 2, 5, 10, 20, 50, 100.
Expansion terms	Number of expansion terms: 2, 5, 10, 15, 20.
Expansion min-docs	Minimal number of documents an expansion term should appear in: 2, 5, 10, 20, 50.

We define a set of 65 features for L2R models on a pair (q_k, c_j) that can be divided into four different groups: 43 features computed using query word statistics (QueryStats), 16 features describing the linguistic properties of the query (QueryLing), 1 feature describing the retrieval model (RetModel), and 4 features related to query expansion parameters (Expansion). The two last groups of features are the same as shown in Table 1. We provide a brief description of QueryStats and QueryLing features.

QueryStats: These features are query-dependent statistical features that were previously used in both query difficulty prediction [4, 6] and learning to rank [9] settings. These features include query terms statistics such as variations of their IDF in the collection (min, max, avg, and sum IDF over the query terms), Query Feedback [14] (min, max, etc.) calculated using various numbers of feedback documents and several default retrieval models such as QL and Bo2, or variant of the NQC which is based on the standard deviation of retrieved documents scores [12].

QueryLing: These are also query-dependent features, but they focus on modelling the linguistic properties of the query. We implemented the features defined in [10], such as the number of WordNet synsets for query terms, the number of prepositions in the query, and so on.

Query-dependent features aim to inform the L2R technique about the characteristics of the query, thus allowing to select different system configuration on a per query basis. As a first investigation, we use all the reasonable features at our disposal without performing any feature selection, leaving this aspect for future work. While the study of the influence of the intrinsic parameters of retrieval models, such as Language Models or BM25, is out of the scope of this paper, we plan to explore it in future experiments. In the following experiments, we used the default parameters provided by Terrier for all the retrieval models.

We use three common performance metrics to rank system configurations: MAP, P@100, and Rprec. These metrics are chosen because they were found to be the least correlated [1].

Three types of L2R techniques have been proposed in the literature based on point-wise, pair-wise and list-wise principles [8]. The point-wise approaches aim at learning to predict a relevance score or class for each document, while the pair-wise approaches learn to predict if one document is more relevant than another. Finally, the list-wise models consider the whole list of documents and optimise a ranking measure. All the L2R models could be suitable to our task: they can rank system configurations in such a way that the best configuration will be ranked first. This is the configuration that we want to select. Notice, however, that the relative positions of the elements at lower positions are also important in L2R models, in particular in pair-wise and list-wise models. The optimisation related to this part of ranking may not be crucial or necessary for our task. Our learning objective could be different. However, we do not examine this question in this paper.

We experimented with a large selection of the existing L2R techniques made available by the RankLib[3] and the SVM^{rank} [4] toolkits. The performance varies largely among the models. Due to space limit, we only report the results with the following representative models: Gradient Boosted Regression Trees (GBRT) [5], Random Forests [3], LambdaMART [13], and SVM^{rank} [7], since we found that these techniques were the most effective for our task. Our selection of L2R techniques covers all the three categories: GBRT and Random Forests are point-wise techniques, SVM^{rank} is pair-wise, and LambdaMART is classified as both pair-wise and list-wise. We trained the three techniques implemented in RankLib to optimise the nDCG@10 of the ranked list of configurations, in which the effectiveness of a configuration is used *in lieu* of the relevance grade of the original L2R algorithms. SVM^{rank} has its own optimisation criteria.

3. EXPERIMENTAL RESULTS

We carried out experiments on two TREC AdHoc test collections: TREC-7 (queries 351-400) and TREC-8 (queries

[1]Impossible configurations, such as using nil expansion method but a number of expansion documents and terms, are excluded.

[2]Details can be found at terrier.org/docs/v4.0/javadoc.

[3]sourceforge.net/p/lemur/wiki/RankLib/

[4]www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Table 2: Results with different L2R models and feature ablations. △ indicates statistically significant improvements over the Grid Search baseline, according to a paired t-test ($p < 0.05$). ▼ indicates statistically significant decreases induced by a feature ablation with respect to the corresponding (All) models.

	MAP		P@100		RPrec	
BM25	0.1942		0.1719		0.2330	
Grid Search	0.2480		0.2213		0.2835	
Random Forests (All)	0.3319 △		0.2785 △		0.3439 △	
- QueryStats	0.3180 △	(-4.17%)	0.2947 △	(+5.80%)	0.3658 △	(+6.35%)
- QueryLing	0.3367 △	(+1.43%)	0.2835 △	(+1.80%)	0.3507 △	(+1.96%)
- RetModel	0.3210 △	(-3.28%)	0.2746	(-1.44%)	0.3462 △	(+0.65%)
- Expansion	0.2201▼	(-33.68%)	0.1843▼	(-33.84%)	0.2384▼	(-30.69%)
SVM^{rank} (All)	0.3073 △		0.2529		0.3204	
- QueryStats	0.2820 △	(-8.23%)	0.2667 △	(+5.48%)	0.3304 △	(+3.12%)
- QueryLing	0.2918 △	(-5.03%)	0.2501	(-1.11%)	0.3498 △	(+9.19%)
- RetModel	0.3118 △	(+1.48%)	0.2628 △	(+3.91%)	0.3400 △	(+6.10%)
- Expansion	0.1723▼	(-43.92%)	0.1203▼	(-52.43%)	0.1914▼	(-40.28%)
GBRT (All)	0.3338 △		0.2803 △		0.3400 △	
- QueryStats	0.3375 △	(+1.11%)	0.2699	(-3.71%)	0.3275 △	(-3.71%)
- QueryLing	0.2982 △	(-10.68%)	0.2908	(+3.75%)	0.3288 △	(-3.31%)
- RetModel	0.3299 △	(-1.17%)	0.2702	(-3.62%)	0.3581 △	(+5.32%)
- Expansion	0.2345▼	(-29.75%)	0.1775▼	(-36.66%)	0.2505▼	(-26.32%)
LambdaMART (All)	0.3271 △		0.2772 △		0.2873	
- QueryStats	0.3272 △	(+0.03%)	0.2705 △	(-2.42%)	0.2692	(-6.28%)
- QueryLing	0.3324 △	(+1.62%)	0.2695 △	(-2.78%)	0.3486 △	(+21.34%)
- RetModel	0.3144 △	(-3.87%)	0.2713 △	(-2.13%)	0.3528 △	(+22.78%)
- Expansion	0.2188▼	(-33.11%)	0.1456▼	(-47.49%)	0.2078▼	(-27.67%)
Upper bound (oracle performance)	0.4136		0.3434		0.4490	

401-450) for which we merged the queries into one set since the document collection is the same. We used a 5-fold cross validation, where each fold has separate training (3/5), validation (1/5), and test sets (1/5). The training queries are used to train L2R models, the validation queries are used to minimise over-fitting, and the test queries are used to evaluate the learned models. We report the average performance on the test queries in Table 2. For each test query, we use the system configuration that has been ranked first by the learned models.

In order to evaluate the effect of each group of features presented in Section 2 (QueryStats, QueryLing, RetModel, Expansion) for selecting the configuration, we perform the following abalation analysis: we remove one group of features at a time, and perform again the learning and testing without the ablated features in order to see the change in retrieval effectiveness. A large decrease in retrieval effectiveness would indicate that the ablated features are deemed important for the learner.

The rows with (All) mean that the models have been learned using the full set of 65 features, while the other rows exhibit results without the ablated group of features. We compare the results of our approach to two baselines: a Grid Search method, which selects the best configuration on a set of training queries (we used both the training and the validation queries here) and uses it on the test queries. This corresponds to the common practice in IR for setting multiple parameters at once. Notice that this configuration is query-independent. For an easy comparison, we also provide the performance of a standard BM25 run (without query expansion), using the default configuration provided by Terrier. We also report in Table 2 the Upper bound of our method, which uses the best possible system configuration for each query (i.e. the oracle performance).

On analysing Table 2, we can make three main observations. Firstly, and most importantly, we see that all L2R techniques can effectively learn to rank reasonable system configurations. All the L2R models can produce much better results than the traditional way (Grid Search) to set system parameters. This result clearly indicates the benefit of using a L2R model to select an appropriate system configuration for a query, rather than setting a unique configuration globally. Among the L2R models, Random Forest, GBRT and LambdaMART produce equivalent performances, while SVM^{rank} performs slightly lower. The superiority of pair-wise and list-wise models over point-wise models cannot be concluded. This observation differs slightly from the traditional use of L2R models where pair-wise and list-wise models are found to perform better than point-wise models [8]. The difference can be explained by the fact that the relative positions of configurations at lower ranks have important impact in pair-wise and list-wise L2R models, while this is not important for our task. The L2R models also compare favorably to the best performing systems of the TREC-7 and TREC-8 AdHoc tracks. The best systems at TREC-7 and TREC-8 achieved 0.3032 and 0.3303 in MAP, while the LambdaMART (All) model can produce 0.3148 and 0.3396 in MAP on the two separate sets of queries.

Secondly, we observe that ablating the EXPANSION group of features always significantly decreases the performance of the learned models, hinting the huge importance of these features for learning an effective model. A possible explanation is that the best parameters for query expansion vary a lot across queries, and the other features are unable to make differences among them. Therefore, in absence of the expansion features, the L2R models cannot make an informed choice on system configuration.

The ablation of the other groups of features has less impact. When we remove the RETMODEL feature, the performances can increase or decrease slightly. This highlights the fact that our approach is able to effectively rank system configuration even without using the retrieval model as a feature. A possible explanation is that the other features are often sufficient to determine the best configuration, or the selections of retrieval model across queries are usually consistent so that the feature about the retrieval model does not provide much additional help. Similar observation also holds on the query-dependent features. However, this does not mean that query-dependent features are useless. There could be more questions about the specific features in this group : some of the features could be redundant, while some others do not seem to be strongly informative. Notice also that we have two groups of query features. When we remove one of them in our ablation analysis, it is possible that the other group can still provide similar information about the query. We will perform more analyses on the features and their effects in future work.

Finally, the results demonstrate the importance of making query-dependent configuration selections. The low impact of query-dependent features in configuration selection does not mean that the final selection of configuration is not query-dependent. For each query, a L2R model always makes a different selection based on the available features. The benefit of query-dependent selection can be best seen by contrasting the L2R models with Grid search, which sets the best query-independent configuration: We can see tat all the query-dependent configuration selections by L2R models are better than Grid search. However, compared to the Oracle performance, we also see that the selections by L2R models are not always the best for each query. There is much room for improvements on this in the future.

4. CONCLUSION

In this paper, we proposed a new approach to set system configuration using learning-to-rank methods. We showed that this is a feasible approach, and it can produce superior performance to the state of the art. Our experiments also showed the importance and benefits to make query-dependent configuration setting. The feature ablation analysis showed various impact of different features. In particular, query features showed lower impact than we expected. More investigations are needed to fully understand the reason. In this study, we did not make a selection of the features to be used and simply used all the features proposed in the literature that sound relevant. However, we observe that the relevance of some of the features to our task may be low. The features can also be redundant, providing similar information. It will be useful to perform feature selection in the future.

This study is a first investigation in the new direction of learning to set system configurations. Many underlying questions remain to be addressed in the future. For example, should the learning objective be different from those used in the L2R algorithms? How can we define a different learning algorithm specifically for ranking configurations?

Acknowledgments

This work is partly supported by the French Agency for Scientific Research (Agence Nationale de la Recherche) under the CAAS project (ANR 2010 CORD 001 01), and partly by an NSERC-Discovery grant.

5. REFERENCES

[1] A. Baccini, S. Déjean, L. Lafage, and J. Mothe. How many performance measures to evaluate information retrieval systems? *Knowledge and Information Systems*, 30(3), 2012.

[2] A. Bigot, S. Déjean, and J. Mothe. Learning to Choose the Best System Configuration in Information Retrieval: the Case of Repeated Queries. *Journal of Universal Computer Science*, 21(13):1726–1745, 2015.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[4] D. Carmel and E. Yom-Tov. Estimating the query difficulty for information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–89, 2010.

[5] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[6] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *Proc. of CIKM*, 2008.

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

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

[9] C. Macdonald, R. L. Santos, I. Ounis, and B. He. About learning models with multiple query-dependent features. *ACM Transations on Information Systems*, 31(3), 2013.

[10] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty. In *Proc. of SIGIR*, 2005.

[11] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proc. of OSIR*, 2006.

[12] A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits. Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems*, 30(2):11:1–11:35, 2012.

[13] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[14] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proc. of SIGIR*, 2007.