Query Performance Prediction Focused on Summarized Letor Features

Adrian-Gabriel Chifu, Léa Laporte, Josiane Mothe, Md Zia Ullah

Query Performance Prediction Focused on Summarized Letor Features

Adrian-Gabriel Chifu
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France
adrian.chifu@univ-amu.fr

Léa Laporte
INSA Lyon, LIRIS, UMR 5205 CNRS
Lyon, France
lea.laporte@insa-lyon.fr

Josiane Mothe
ESPE, Université de Toulouse, IRIT, UMR5505 CNRS
Toulouse, France
Josiane.Mothe@irit.fr

Md Zia Ullah
UPS, Université de Toulouse, IRIT, UMR5505 CNRS
Toulouse, France
mdzia.ullah@irit.fr

ABSTRACT

Query performance prediction (QPP) aims at automatically estimating the information retrieval system effectiveness for any user's query. Previous work has investigated several types of pre- and post-retrieval query performance predictors; the latter has been shown to be more effective. In this paper we investigate the use of features that were initially defined for learning to rank in the task of QPP. While these features have been shown to be useful for learning to rank documents, they have never been studied as query performance predictors. We developed more than 350 variants of them based on summary functions. Conducting experiments on four TREC standard collections, we found that Letor-based features appear to be better QPP than predictors from the literature. Moreover, we show that combining the best Letor features outperforms the state of the art query performance predictors. This is the first study that considers such an amount and variety of Letor features for QPP and that demonstrates they are appropriate for this task.

KEYWORDS

Query performance prediction, Query difficulty prediction, Query features, Post retrieval features, Letor features

https://doi.org/10.1145/3209978.3210121

1 INTRODUCTION

While Information Retrieval system effectiveness is usually measured as an average effectiveness over queries, it is well-known that a system performs differently on each individual queries. *Query performance prediction* (QPP) aims at automatically estimating the information retrieval system effectiveness for any user's query difficulty without relevance judgement [7]. Predicting query performance is a first mandatory step to be able to adapt query processing when the query is detected as difficult. System adaptation can consist in selective query expansion [2], or selecting the best system configurations depending on the query features [9], for instance. Thus, having accurate predictors is a hot and very important topic.

The literature of the field has investigated the use of various types of predictors. Pre-retrieval features can be calculated from the query itself. IDF is an example of a pre-retrieval feature which is calculated by extracting the IDF of each query term from the inverted file and then aggregating the values over the query terms [11]. Some pre-retrieval features need external resources to be calculated as for example the number of query term senses based on WordNet [15]. On the other hand, calculating post-retrieval features requires evaluating the query over the whole collection of documents. They are usually based on functions applied to the document scores or retrieved document lists. For example, Query Feedback (QF) measures the overlap between two retrieved document lists for the original query and the expanded query [22]. Although post-retrieval features are based on retrieved document scores, they are not Letor features strictly speaking. Indeed, Letor features have been first used in learning to rank applications where they have been shown to be effective [4, 6, 17, 19]. Letor features have also been used in other applications such as learning to rank system configurations [9] or reducing verbose textual queries [3], but have never been used as QPP. A systematic review does not yet exist in the context of QPP; this is the purpose of this short paper.

Recent work has focused on ways of combining predictors rather than finding better predictors [10, 18, 22]. Although combining features has been shown to be effective, we believe that the better the features, the better the QPP. Our paper revisits this perspective by evaluating more than 350 post-retrieval features (based on variants of Letor-based features) as QPP features. The paper focuses on the post-retrieval features since they have been shown to be more effective than pre-retrieval features for QPP [21].

2 SUMMARIZED LETOR-BASED FEATURES

Letor features have been widely used in the context of learning to rank [17]. The main goal of learning to rank is to learn a function to better rank retrieved documents given a query; which is a different problem than QPP. Terrier's FAT component [14] implements most of the LETOR feature that can be found in [17][1] as well as a few others[2]. Most of the features are calculated from a matching score between the query and the document, either considering the document title or the full content, for a total of 39 features.

All the features are associated with query-document pairs. To make the Letor features (LF) usable as query features, we have used different summary functions over the retrieved documents, for a given query. More formally, let q be a query in the set of all queries Q and $\mathbb{D}_{q,n}$ be the set of the top-ranked n retrieved documents for the query q. We compute the i-th summarized Letor feature, $\mathsf{SLF}_i(\mathbb{S}, q)$ as follows:

$$\mathsf{SLF}_i(\mathbb{S}, q) = \varphi_{\mathbb{S}}(\{\mathrm{LF}_i(d, q)\}, q, \mathbb{D}_{q,n})$$

where $d \in \mathbb{D}_{q,n}$ is a document, $\{\mathrm{LF}_i(d, q)\}$ is the set of values for the Letor feature i for each couple (d, q) and $\varphi_{\mathbb{S}}$ is a summary function with $\mathbb{S} \in \{Min, Max, Mean, Q1, Median, Q3, Std, Var, and\ Sum\}$.

We used 9 summary functions for each LF:

- $\varphi_{Min}, \varphi_{Max}, \varphi_{Mean}, \varphi_{Sum}$: the *minimum* (*maximum, average, and sum*) value of the Letor feature over the retrieved documents;
- $\varphi_{Q1}, \varphi_{meadian}, \varphi_{Q3}$: after calculating the Letor feature for each retrieved document, we sorted the values in increasing order; then the set is divided into quartiles. $Q1$ (respectively median, $Q3$) is the value that makes at least one quarter (2 quarters, 3 quarters) of the values having a score lower than $Q1$ (respectively median, $Q3$);
- $\varphi_{Std}, \varphi_{Var}$: the *standard deviation* and the *variance* of the Letor feature values.

Each of the summary functions leads to a variant of the Letor feature. Thus, we have a group of summarized Letor features (SLF) for each LF. As opposed to previous research that analyses a few features, in this study, we analysed more than 350 SLF.

3 EVALUATION OF THE FEATURES

The accuracy of a query performance predictor is usually measured in the literature by evaluating the link (e.g. correlation value) between the predictor values (considered as the predicted effectiveness) and the actual system effectiveness values [11, 18]. Pearson correlation is usually considered; it assumes a linear correlation between the two analyzed variables. Since correlation may exist without being linear, we complete the study using Kendall correlation as well.

The results can be biased by the choice of the system used as a reference. What is now a common reference in QPP is Language Modeling with Dirichlet smoothing and μ=1000 [8, 18, 20, 22]. We also focus on this system in this paper and keep the analysis of system dependency for future work. With regard to system effectiveness, we considered AP and NDCG, which is also common practice in IR evaluation. Finally, we use four reference collections from TREC: Robust (528,155 newspaper documents, 250 topics), WT10G (1.6 million web/blog documents, 100 topics), GOV2 (25 million web documents, 150 topics), and ClueWeb12B (52 million web documents, 100 topics).

3.1 Individual effectiveness

We studied the individual effectiveness for QPP of any single summarized Letor feature. We also considered the state of the art QPP features, namely: Clarity [8], QF [22], WIG [22], UQC [21], and NQC [21].

Since we have more than 350 SLF, it was not possible to present all the correlations; we selected the most correlated features. For each collection, we selected the top 4 features and reported their correlation for the 4 collections in Table 1. In the first part of the table, we present AP while the second part reports NDCG (both @1000). The features are ordered according to decreasing Pearson correlation on GOV2.

In Table 1, we can see that many of the SLF are more correlated than state of the art QPP features, at least on one of the collections, but many times across several collections. The most correlated feature (AP and NDCG) is *WMODEL.DFIZ_std* which represents a weighting model based on the standardized distance from independence in term frequency [16]. *WMODEL.ML2_std* (second feature for AP) represents a weighting model based on multinomial randomness model, with Laplace after-effect model and normalisation 2 [12]. *WMODEL.In_expC2_max* (second feature for NDCG) is the Inverse Expected Document Frequency weighting model with Bernoulli after-effect and normalization [1]. When considering the 4 different collections, the 2 correlation measures, the 2 evaluation measures, and all the SLFs, we found out that several of them consistently correlate significantly across collections and measures. This is the case for the first 5 SLFs in Table 1. None of the state of the art features are as robust across collections and measures. The state of the art feature that has a similar robustness is QF, but it is not significantly correlated for AP on ClueWeb12B. We also observed that the highest correlations are with SLF calculated on document content rather than on part of it such as the title only.

3.2 The impact of n

In the previous section, the SLF have been calculated when considering the n (1000) top-ranked retrieved documents. However, the value of n may have an important impact on the correlation [21]. It was not possible to report the effect of n on all the SLF. Since the idea was to observe the influence of n if any, we chose the first SLF from Table 1, namely *WMODEL:DFIZ_std*. It corresponds to the feature from Terrier calculated using Divergence From Independence model based on Standardization [13], to which we applied standard deviation when summarizing the values across documents and we showed it is robust across collections for QPP.

Table 2 reports the Pearson correlation that we obtained on the four collections. We can see that the best value of n is not consistent across collections while it is consistent for AP and NDCG. GOV2 requires fewer documents (top 500) to get the highest correlation than the other collections. This result holds also for other SLFs we analysed. An interesting track for future work is to analyse whether the optimal number of documents is the same across features or not. The robustness of the number of documents across queries is also an interesting problem to study.

[1] Features name are the ones used in Terrier; they correspond to different implementations of features number 3, 5, 13, 15, 20, 23, 25, 33, and 35 in Table 2 from [17].
[2] All the features are detailed at http://www.terrier.org/docs/v4.0/javadoc/index.html?org/terrier/matching/models/WeightingModel.html

Table 1: Pearson (P.) and Kendall (K.) correlations between the features and the actual system effectiveness (AP@1000 and NDCG@1000). The top four correlated features are selected from each collection separately and then all are ordered by decreasing order based on Pearson correlation on GOV2. We also included the features from the literature that are not Letor-based. "*" stands for p-value < 0.05, while "+" stands for p-value < 0.001. Values in bold are higher than QF baseline.

	Robust		WT10G		GOV2		ClueWeb12B	
AP	**P.**	**K.**	**P.**	**K.**	**P.**	**K.**	**P.**	**K.**
WMODEL.DFIZ_std	.191*	.200+	.217*	**.235+**	**.453+**	**.305+**	**.298***	**.255+**
WMODEL.ML2_std	.251+	.220+	.216*	**.290+**	**.453+**	**.328+**	**.265***	**.256+**
WMODEL.In_expC2_max	.320+	**.359+**	**.403+**	**.298+**	**.449+**	**.306+**	**.303***	**.225+**
WMODEL.PL2_std	.334+	.296+	.211*	**.297+**	**.449+**	**.319+**	**.267***	**.255+**
WMODEL.In_expB2_max	.316+	**.353+**	**.389+**	**.296+**	**.438+**	**.297+**	**.304***	**.228+**
WMODEL.IFB2_max	.318+	**.353+**	**.399+**	**.304+**	**.437+**	**.293+**	**.298***	**.219***
WMODEL.BB2_max	.298+	**.344+**	**.374+**	**.277+**	**.436+**	**.300+**	**.308***	**.228+**
WMODEL.IFB2_std	.393+	.337+	.318*	**.310+**	.394+	.260+	**.231***	**.232+**
WMODEL.DFReeKLIM_max	.343+	.287+	.313*	**.202***	.385+	.277+	**.376+**	**.259+**
WMODEL.DPH_max	.347+	.292+	.323*	**.223***	.385+	.276+	**.374+**	**.253+**
WMODEL.Js_KLs_max	.305+	.260+	.266*	.164*	.367+	.262+	**.379+**	**.253+**
WMODEL.DFRee_max	.295+	.251+	.252*	.152*	.359+	.255+	**.377+**	**.249+**
WMODEL.SFM.DirichletLM_max	.392+	.324+	.235*	**.210***	.350+	.270+	**.263***	**.163***
WMODEL.LemurTF_IDF_max	.414+	**.348+**	.342+	**.282+**	.298+	.251+	**.223***	**.186***
WMODEL.LemurTF_IDF_std	**.457+**	**.356+**	.207*	**.279+**	.258*	.230+	**.170**	**.207***
QF [22]	*.432+*	*.343+*	*.343+*	*.169**	*.407+*	*.285+*	*.163*	*.114*
Clarity [8]	.435+	.308+	.239*	.189*	.112	.086	-.148	-.108
WIG [22]	.315+	.253+	.181	.122	.302+	.239+	.366+	.201*
UQC [21]	.496+	.358+	.311*	.192*	.286+	.191+	.152	.064
NQC [21]	.125*	.203+	.010	.066	-.037	.086	-.095	-.047
NDCG	**P.**	**K.**	**P.**	**K.**	**P.**	**K.**	**P.**	**K.**
WMODEL.DFIZ_std	.306+	.235+	.321*	**.261+**	**.447+**	**.334+**	**.301***	**.239+**
WMODEL.In_expC2_max	.338+	**.359+**	**.429+**	**.279+**	**.444+**	**.326+**	**.264***	**.200***
WMODEL.DFIC_std	.291+	.221+	.294*	.221*	**.441+**	**.328+**	**.303***	**.241+**
WMODEL.ML2_std	.326+	.248+	.290*	**.294+**	**.438+**	**.359+**	**.264***	**.230+**
WMODEL.IFB2_max	.334+	**.351+**	**.420+**	**.282+**	**.429+**	**.312+**	**.257***	**.195***
WMODEL.PL2_max	.434+	.329+	**.389+**	**.258+**	**.412+**	**.322+**	**.324+**	**.235+**
WMODEL.DPH_std	.380+	.296+	**.432+**	**.342+**	.397+	**.308+**	**.238***	**.195***
WMODEL.Js_KLs_max	.364+	.271+	.280*	.161*	.382+	.289+	**.361+**	**.243+**
WMODEL.BM25_std	.428+	.338+	.320*	**.295+**	.344+	.289+	.162	**.174***
WMODEL.DFR_BM25_std	.429+	.339+	.324*	**.298+**	.341+	.290+	.160	**.173***
WMODEL.XSqrA_M_Q3	-.055	-.040	-.030	-.032	.335+	.217+	**.363+**	**.245+**
WMODEL.XSqrA_M_mean	-.066	-.047	-.059	-.046	.308+	.202+	**.359+**	**.235+**
WMODEL.SFM.DirichletLM.._std	.354+	.280+	**.430+**	**.253+**	.302+	.215+	.200*	**.172***
WMODEL.LemurTF_IDF_std	.440+	.340+	.206*	**.267+**	.243*	.247+	.114	.151*
QF [22]	.504+	.349+	.350+	.223*	.398+	.297+	.210*	.152*
WIG [22]	.364+	.262+	.228*	.149*	.329+	.273+	.342+	.186*
UQC [21]	.436+	.343+	.304*	.226+	.285+	.213+	.083	.020
Clarity [8]	.452+	.308+	.255*	.170*	.111	.100	-.132	-.099
NQC [21]	.117	.191+	.065	.106	-.004	.091	-.143	-.069

3.3 Combining features for QPP

Assuming that different QPP features capture diverse information about query performance, whether a combination of features is likely to be a better predictor than a single feature. Following this assumption, a few studies have investigated the linear combination of features [5, 10, 20, 22]. These studies combine less than 10 features.

Considering that we have investigated more than 350 SLFs in the individual effectiveness analysis, using all these features in a single model such as linear regression would not be optimum. In this paper, we choose to analyse the linear combination of the features that correlate the most with the effectiveness measure to predict, for each collection. Thus, for each collection, we plotted the correlations values for the best 20 features in decreasing order, then empirically searched for a gap in the correlation values, by looking at the differences between consecutive correlation values, two by two. We thus selected all the features before this gap. Note that the features may be different depending on whether we consider

Table 2: Pearson correlation of the WMODEL:DFIZ_std [13] SLF predictor according to n, the number of top-ranked retrieved documents considered when calculating the feature.

	n	5	10	50	100	500	1000
AP	Robust	.056	.065	.089	.081	.146*	**.191***
	WT10G	.027	-.084	.163	.142	.175	**.217***
	GOV2	.261*	.385+	.409+	.407+	**.453+**	.453+
	ClueWeb12B	.320*	.255*	.266*	.243*	.269*	**.298***
NDCG	Robust	.081	.119	.183*	.191*	.252+	**.306+**
	WT10G	.069	-.032	.224*	.217*	.294*	**.321***
	GOV2	.285+	.397+	.426+	.418+	**.453+**	.447+
	ClueWeb12B	.246*	.234*	.253*	.224*	.265*	**.301***

correlation on AP or NDCG. For AP, this process selected 5 features per collection, except for ROBUST (6 features). For NDCG, using the same process we selected 6 features for WT10G and ClueWeb12B, 8 features for Robust, and 10 features for GOV2. We leave the investigation of other feature selection methods such as LASSO (more sparse) and other machine learning methods such as random forest for future work since our main topic here is to investigate the effectiveness of new predictors.

Similarly to [22] and [10], we used multiple linear regression to combine the features into a single performance prediction model. We considered five different sets of predictors to be combined. A first set S_1 contains the two state of the art predictors *WIG* and *QF*. It corresponds to our baseline. The second set S_2 is composed of the two best Letor predictors on a given collection, according to the simple linear regression experiment. This set can be easily compared to the baseline since it is also composed of two features (thus the same level of complexity and costs to be calculated). The third set S_3 contains all the best LETOR features per collection we selected as explained previously. The fourth set S_4 is the union of sets S_1 and S_2, thus it combines the state of the art and the two best SLF features per collection. Finally, we considered all the best predictors selected per collection together with the two baselines *WIG* and *QF* in the fifth set S_5. We considered the same collections, system, effectiveness, and evaluation measures as in the individual performance prediction analysis. We performed the leave-one-out cross-validation to predict query performance. As shown in Table 3,

Table 3: Performance of linear regression of combined post-retrieval predictors according to Pearson correlation.

	Combination	**Robust**	**WT10G**	**GOV2**	**ClueWeb**
AP	S_1: WIG + QF	.459+	.274*	.399+	.287*
	S_2: 2 Best SLF	.382+	.404+	.438+	.237*
	S_3: Best SLF	.402+	.339+	.420+	**.302***
	S_4: $S_1 \cup S_2$	**.478+**	.420+	**.465+**	.260*
	$S_1 \cup S_3$: All	.454+	**.427+**	.509+	.208*
NDCG	S_1: WIG + QF	.537+	.303*	.405+	.286*
	S_2: 2 Best SLF	.430+	.449+	.469+	.211*
	S_3: Best SLF	.458+	.457+	.464+	**.312***
	S_4: $S_1 \cup S_2$	**.556+**	**.468+**	**.514+**	.293*
	S_5: $S_1 \cup S_3$	.526+	.446+	.487+	.188

for all the collections except ClueWeb12B and NDCG, the best correlation values are obtained when considering a combination of the two best Letor features and the two state of the art features. Considering ClueWeb12B and NDCG, the best correlation value is obtained when considering a linear combination of the best Letor features. If we take a fine-grained look at the results, we observe that the best results are obtained when considering either the Letor features alone or combined with WIG and QF, whatever the effectiveness measure is. As an additional analysis, we computed the correlations between the features from the literature and the best Letor predictors. The correlations values were not significant, supporting our finding that Letor features are complementary to state-of-the-art ones. Finally, with regard to AP, results are not directly comparable to Raiber's [18] (they found Pearson correlation of .557 for Robust, .346 for WT10G, .570 for GOV2) since we could not reproduce their results; for example, the Pearson correlation values they reported for individual state of the art QPPs are higher than the ones we re-calculated using the same collections and the same reference system.

4 CONCLUSION

We showed that the Summarized Letor features we defined are good query performance predictors and that some of them are more robust than the ones from the literature across collections. Moreover, we found that they are complementary to the existing features as when combined we achieved better QPP.

REFERENCES

[1] Giambattista Amati. 2003. Probability Models for IR based on Divergence from Randomness. *Department of CS* PhD (2003), 182.
[2] G. Amati, C. Carpineto, and G. Romano. 2004. Query difficulty, robustness, and selective application of query expansion. In *ECIR*. 127–137.
[3] Jaime Arguello, Sandeep Avula, and Fernando Diaz. 2017. Using Query Performance Predictors to Reduce Spoken Queries. In *ECIR*. Springer, 27–39.
[4] Niranjan Balasubramanian and James Allan. 2010. Learning to select rankers. In *ACM SIGIR*. 855–856.
[5] Shariq Bashir. 2014. Combining pre-retrieval query quality predictors using genetic programming. *Applied Intelligence* 40, 3 (2014), 525–535.
[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*. 129–136.
[7] David Carmel and Elad Yom-Tov. 2010. Estimating the query difficulty for IR. *Syn. Lect. on Inf. Concepts, Retrieval, and Services* (2010), 1–89.
[8] Steve Cronen-Townsend and W. Bruce Croft. 2002. Quantifying Query Ambiguity. In *Conference on Human Language Technology Research*. 104–109.
[9] Romain Deveaud, Josiane Mothe, and Jian-Yun Nie. 2016. Learning to rank system configurations. In *CIKM*. ACM, 2001–2004.
[10] Claudia Hauff. 2010. Predicting the effectiveness of queries and retrieval systems. *SIGIR Forum* 44, 1 (2010), 88. https://doi.org/10.1145/1842890.1842906
[11] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. 2008. A survey of pre-retrieval query performance predictors. In *ACM CIKM*. 1419–1420.
[12] İlker Kocabaş, Bekir Taner Dinçer, and Bahar Karaoğlan. 2014. A Nonparametric Term Weighting Method for Information Retrieval Based on Measuring the Divergence from Independence. *Inf. Retr.* 17, 2 (April 2014), 153–176.
[13] İlker Kocabaş, Bekir Taner Dinçer, and Bahar Karaoğlan. 2014. A nonparametric term weighting method for information retrieval based on measuring the divergence from independence. *Information retrieval* 17, 2 (2014), 153–176.
[14] Craig Macdonald, Rodrygo LT Santos, Iadh Ounis, and Ben He. 2013. About learning models with multiple query-dependent features. *TOIS* 31, 3 (2013).
[15] Josiane Mothe and Ludovic Tanguy. 2005. Linguistic features to predict query difficulty. In *Predicting query difficulty, ACM SIGIR workshop*. 7–10.
[16] Vassilis Plachouras and Iadh Ounis. 2007. Multinomial Randomness Models for Retrieval with Document Fields. In *ECIR*. 28–39.
[17] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for IR. *Information Retrieval* 13, 4 (2010), 346–374.
[18] Fiana Raiber and Oren Kurland. 2014. Query-performance prediction: setting the expectations straight. In *ACM SIGIR*. ACM, 13–22.
[19] D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. 2011. LambdaMerge: merging the results of query reformulations. In *ACM WSDM*. 795–804.
[20] Anna Shtok, Oren Kurland, and David Carmel. 2010. Using statistical decision theory and relevance models for QPP. In *ACM SIGIR*. 259–266.
[21] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting query performance by query-drift estimation. *ACM TOIS* 30, 2 (2012).
[22] Yun Zhou and W Bruce Croft. 2007. Query performance prediction in web search environments. In *ACM SIGIR*. 543–550.