Combining Word Embedding Interactions and LETOR Feature Evidences for Supervised QPP

Suchana Datta[1], Debasis Ganguly[2], Josiane Mothe[3,4] and Md Zia Ullah[5]

[1] University College Dublin, Ireland
[2] University of Glasgow, UK
[3] Université de Toulouse, INSPE, UT2J, Toulouse, France
[4] IRIT CNRS, UMR5505, Toulouse, France
[5] Edinburgh Napier University, UK

Abstract
In information retrieval, query performance prediction aims to predict whether a search engine is likely to succeed in retrieving potentially relevant documents to a user's query. This problem is usually cast into a regression problem where a machine should predict the effectiveness (in terms of an information retrieval measure) of the search engine on a given query. The solutions range from simple unsupervised approaches where a single source of information (e.g., the variance of the retrieval similarity scores in NQC), predicts the search engine effectiveness for a given query, to more involved ones that rely on supervised machine learning making use of several sources of information, e.g., the learning to rank (LETOR) features, word embedding similarities etc. In this paper, we investigate the combination of two different types of evidences into a single neural network model. While our first source of information corresponds to the semantic interaction between the terms in queries and their top-retrieved documents, our second source of information corresponds to that of LETOR features.

Keywords
Query performance prediction, CNN, Feature combination, Word embedding, LETOR features

1. Introduction

Query performance and query difficulty predictions are two sides of the same coin. For both, the objective is to predict whether the search engine is likely to succeed in the task of retrieving relevant documents to the user's query. More precisely, for query difficulty prediction, the problem is generally cast into a classification problem. A difficult query is then a query for which the search engine is poorly performing considering an effectiveness measure. The notion of *difficult* or *hard* queries as opposed to *easy* queries is used in [1, 2]. More than two classes were also used in related work [3, 4] where different definitions of query difficulty are introduced.

QPP++ 2023: Query Performance Prediction and Its Evaluation in New Tasks, co-located with 45th European Conference on Information Retrieval (ECIR) from the 2nd to the 6th of April 2023 in Dublin, Ireland
suchana.datta@ucdconnect.ie (S. Datta); debasis.ganguly@glasgow.ac.uk (D. Ganguly); Josiane.Mothe@irit.fr (J. Mothe); m.ullah@napier.ac.uk (M. Z. Ullah)
0000-0001-9220-6652 (S. Datta); 0000-0003-0050-7138 (D. Ganguly); 0000-0001-9273-2193 (J. Mothe); 0000-0002-4022-7344 (M. Z. Ullah)

On the other hand, query performance prediction (QPP) aims at estimating the effectiveness of a search performed on a query by a search engine without document relevance judgement [5, 6, 7, 8]. It is usually cast into a regression problem where a machine should predict the effectiveness (in terms of a measure) of the search engine on a given query. The solutions go from very simple machines where a single feature predicts the search engine effectiveness, to very sophisticated machine relying on supervised machine learning [9, 10, 11] and implying many criteria. For single features, Hauff et al. [12] surveyed 22 pre-retrieval features from the literature at that time, from which some are computed considering the query itself only, other use information from the indexed collection. As opposed to pre-retrieval features, post-retrieval features imply that a first document retrieval is run using the query before the feature can be calculated; post-retrieval features make use of the document scores [13, 14, 15, 16, 17, 18]. Several features were combined in more complex predictive models such as SVM and Decision trees [19], Genetic algorithms [20], Feature selection models [8], Linear combination [21, 22] or Neural networks [18, 23].

Queries can also be ordered in terms of the predicted effectiveness of the search engine, or it is possible to know among two queries which of the two is predicted as easier that the other. Datta et al. [23] defined a neural model that learns this relationship. It captures term semantics and interactions at the query and top/bottom-retrieved document levels. Their model also includes the prediction of effectiveness for a given query. On the other hand, Chifu et al. [21] have shown that LETOR features that were initially defined for learning to rank documents [24] are good indicators for QPP. Examples of LETOR features are the BM25 score of the document with regards to the query, the frequency of the query terms appearing in the document, and the PageRank score of the document [24]. In this paper, we investigate the combination of the two types of evidences into a single model. The network input in Datta et al. [23] is a flattened 3D matrix of term embeddings for (query, document) pairs while LETOR features can be calculated for (query, document) pairs or can be aggregated at the query level [21].

2. Model

2.1. Query representation

Following the model presented in [23], as a first step, the idea is to calculate the cosine similarities between the embedded representation of terms of the query Q_a and embedded representation of terms of the document $D_i^a : D_i^a \in R(Q_a)$ where $R(Q_a)$ is the set of retrieved documents considered for the interaction with Q_a consisting of t top and b bottom retrieved documents. Similar to [25], the distribution of similarities between the j^{th} query term q_j and the D_i^a terms is then transformed into a vector of fixed length p by computing a histogram of the similarity values over a partition of p equi-spaced intervals defined over the range of these values (i.e., the interval $[-1, 1)$) (see [23] for details). For each query, we obtain a matrix of (t+b) vectors of dimension p which model the semantic interaction between a query and the retrieved documents for that query. This results into a 3^{rd} order interaction tensor that we denote $Q_a \oplus R(Q_a) \in \mathbb{R}^{(t+b)\times k\times p}$.

Here, we consider both word embedding information and the LETOR post-retrieval features as in [21]. In learning to rank documents, LETOR features indicate how relevant or important the document is with respect to the query [24]. Similar to semantic interaction vectors, LETOR features are associated with a query-document pair. We thus can concatenate this 3^{rd} order

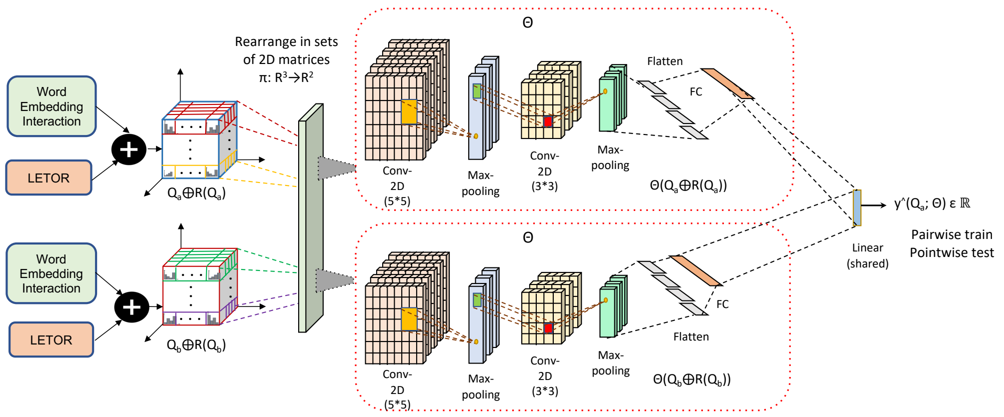

Figure 1: The end-to-end QPP model consists of a siamese network of shared parameters of layered convolutional feature extraction, followed by a linear activation layer with pairwise hinge loss for pointwise testing yielding a score for a given query.

LETOR tensor with the previous interaction tensor which then operates at the early stage.

2.2. Deep Model for QPP

To extract convolutional features from the 3^{rd} order interaction tensor, $Q_a \oplus R(Q_a)$, we first need to slice the 3^{rd} order tensor into separate matrices (2^{nd} order tensors), on each of which, 2D convolution can be applied to extract distinguishing features from the raw data of query-document interactions. Here, we choose to adapt the SDMQ model from [23] which stands for Separate Documents Merged Query-terms. It considers every interaction vector between the j^{th} query term and i^{th} document (see Equation 1) as a separate candidate for convolutional feature extraction. Each such interaction vector between a query-term and a document is of dimension p and there are a total of $(t+b) \times k$ such vectors. We apply 2D convolution on these vectors (See Figure 1).

The β^{th} component ($\beta = 1, \dots, p$) of this interaction vector is given by the count of how many terms yield similarities that lie within the β^{th} partition of $[-1, 1)$ [23], i.e.,

$$(q_j \oplus D_i^a)_\beta = \log(\frac{N_0}{n(q_j)}) \sum_{w \in D_i^a} \mathbb{I}[\frac{2(\beta-1)}{p} - 1 \leq \frac{\vec{q}_j \cdot \vec{w}}{|\vec{q}_j||\vec{w}|} < \frac{2\beta}{p} - 1], \tag{1}$$

where $n(q_j)$ denotes the number of documents in the collection where the j^{th} query term q_j occurs, and N_0 denotes the total number of documents in the collection.

We then cast the QPP problem into a two-step one: first considering two queries, the model should predict which one is easier than the other one; then considering the partial order that results from the first step, it should predict the performance of any of the queries. To solve this problem, we consider an end-to-end model which consists of a siamese network: two networks as presented in Figure 1 are used to process two queries, followed by a linear activation layer.

The network in Figure 1 is trained with instances of query pairs and their labels. Given a training set of queries $\mathcal{Q} = \{Q_1, \dots, Q_m\}$, we construct the set of all unordered pairs of the form (Q_a, Q_b), where $\forall a, b \leq m$ and $b > a$. The reference label, $y(Q_i, Q_j)$, of a paired instance is determined by a relative comparison of the retrieval effectiveness obtained by a system with a target evaluation measure (e.g., average precision). The ground truth is computed using the relevance assessments. It is calculated as $y(Q_a, Q_b) = \mathrm{sgn}(\mathcal{M}(Q_a; \mathcal{R}(Q_a)) - \mathcal{M}(Q_b; \mathcal{R}(Q_b)))$, where $\mathcal{M}$ is an evaluation measure which depends on a set of relevant documents - $\mathcal{R}(Q)$ for a query $Q \in \mathcal{Q}$ and the set of retrieved documents $\mathcal{A}$, $\mathrm{sgn}(x) = 0$ if $x \leq 0$ or 1 otherwise.

With pairwise hinge loss for pointwise testing yielding a score for a given query as in [23]: $\mathcal{L}(Q_a, Q_b) = \max(0, 1 - \mathrm{sgn}(y(Q_a, Q_b) \cdot (\hat{y}(Q_a; \Theta) - \hat{y}(Q_b; \Theta))))$ where $\hat{y}(Q; \Theta)$ is a real-valued score predicted by a linear activation unit from the output of the shared layer of parameters. It is a function of one query rather than a pair.

3. Evaluation framework

3.1. Dataset

To evaluate our proposed model, we would consider TREC6,7,8 and Robust collections, which include 250 topics. As train-test splits, we would consider TREC6,8 and Robust collections as the training set (200 topics), and TREC7 (50 topics) as the testing set. Indeed, the pairwise combination of topics generates 39,402 training and 2,450 testing instances.

3.2. LETOR Features

We rely on [21] regarding LETOR features extraction and aggregation. LETOR feature were extracted from the initial documents retrieved based on the reference configuration using the BM25 model [26]. The query-document features (Terrier's weighting model[1]) are: WMODEL:Tf, WMODEL:TF_IDF, WMODEL:LemurTF_IDF, WMODEL:BM25, WMODEL:Js_KLs, WMODEL:In_expC2, WMODEL:InB2, WMODEL:DLH, WMODEL:ML2, WMODEL:BB2, WMODEL:DFIC, WMODEL:IFB2, WMODEL:InL2, WMODEL:PL2, WMODEL:LGD, WMODEL:MDL2, WMODEL:DirichletLM, WMODEL:DFRee, and WMODEL:Hiemstra_LM.

For aggregated LETOR features, the mean, standard deviation, and maximum summary functions were used to obtain the feature vectors that represent each query-configuration pair. Chifu *et al.* showed that the features obtained with these aggregation functions are complementary for query performance prediction [21].

3.3. Baselines

Our approach should be compared with several unsupervised QPP approaches such as WIG [27], NQC [28], and UEF [29].

WIG [27] As its specificity measure, weighted information gain (WIG) uses the aggregated value of the information gain with each document (with respect to the collection) in the top-retrieved set. The more topically distinct a document is from the collection, the higher its gain will be. Hence, the average of these gains characterizes how topically distinct is top-retrieved set.

[1]http://terrier.org/docs/v5.2/javadoc/org/terrier/matching/models/WeightingModel.html

NQC [28] Normalized query commitment (NQC) estimates the specificity of a query as the standard deviation of the RSV's of the top-retrieved documents with the assumption that a lower deviation from the average (indicative of a flat distribution of scores) is likely to represent a situation where the documents at the very top ranks are significantly different from the rest.
UEF [29] The UEF method assumes that information from some top-retrieved set of documents are more reliable than others. As a first step, the UEF method estimates how robust is a set of top-retrieved documents by checking the relative stability in the rank order before and after relevance feedback (by RLM). The higher the perturbation of a ranked list post-feedback for a query, the greater is the likelihood that the retrieval effectiveness of the initial list was poor.

3.4. Evaluation metric

Pearson's-r and Kendall's-τ that measure the correlation between the predicted effectiveness value and the ground-truth effectiveness are commonly used for QPP evaluation [18, 23]. To measure the ground-truth effectiveness of the system, AP@100 and nDCG@20 are commonly adopted in related works [18, 23].

3.5. Experimental setting

We could follow the pairwise training and pointwise testing framework. The following combinations of interaction and LETOR features with the Neural network model could be experimented:

- **EXP 1 (LETOR):** Making use of LETOR features only for query-document pairs.
- **EXP 2 (SDMQ):** Only interaction features for query-document pairs.
- **EXP 3 (SDMQ + LETOR):** Early combination of interaction and LETOR features for query-document pairs.

4. Discussion and Conclusion

In this paper, we suggest the combination of two approaches that have been developed in two different IR groups. We presented the model as well as the way it could be evaluated. Our preliminary results show that the combination could be effective. For example, when considering the NQC, WIG and UEF baselines, the correlation measures are in the range of 0.21 to 0.34. When considering the proposed combined models, the correlation is up to 0.70 (AP@100 Pairwise accuracy, SDMQ+LETOR).

In future work, we would like to complete the evaluation and analyse the results deeper. We also would like to compare early fusion and late fusion of LETOR feature based model [21] and the one from Datta et al. [23].

References

[1] E. Yom-Tov, S. Fine, D. Carmel, A. Darlow, Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval, in: ACM SIGIR, 2005, pp. 512–519.
[2] C. Lioma, B. Larsen, H. Schutze, User perspectives on query difficulty, in: Advances in Information Retrieval Theory, Springer, 2011, pp. 3–14.

[3] J. Mothe, L. Laporte, A.-G. Chifu, Predicting query difficulty in ir: impact of difficulty definition, in: KSE, IEEE, 2019, pp. 1–6.

[4] C. Macdonald, N. Tonellotto, I. Ounis, On single and multiple representations in dense passage retrieval, arXiv preprint arXiv:2108.06279 (2021).

[5] B. He, I. Ounis, Inferring query performance using pre-retrieval predictors, in: International symposium on string processing and information retrieval, Springer, 2004, pp. 43–54.

[6] Y. Zhao, F. Scholer, Y. Tsegay, Effective pre-retrieval query performance prediction using similarity and variability evidence, in: ECIR, 2008, pp. 52–64.

[7] O. Zendel, A. Shtok, F. Raiber, O. Kurland, J. S. Culpepper, Information needs, queries, and query performance prediction, in: ACM SIGIR, 2019, pp. 395–404.

[8] S. Déjean, R. T. Ionescu, J. Mothe, M. Z. Ullah, Forward and backward feature selection for query performance prediction, in: ACM SAC, 2020, pp. 690–697.

[9] S. Datta, S. MacAvaney, D. Ganguly, D. Greene, A 'pointwise-query, listwise-document' based query performance prediction approach, in: ACM SIGIR, 2022, pp. 2148–2153.

[10] N. Arabzadeh, F. Zarrinkalam, J. Jovanovic, E. Bagheri, Neural embedding-based metrics for pre-retrieval query performance prediction, in: European Conference on Information Retrieval, Springer, 2020, pp. 78–85.

[11] H. Hashemi, H. Zamani, W. B. Croft, Performance prediction for non-factoid question answering, in: ACM SIGIR, 2019, p. 55–58.

[12] C. Hauff, D. Hiemstra, F. de Jong, A survey of pre-retrieval query performance predictors, in: ACM CIKM, 2008, pp. 1419–1420.

[13] S. Cronen-Townsend, Y. Zhou, W. B. Croft, Predicting query performance, in: ACM SIGIR, 2002, pp. 299–306.

[14] A. K. Sehgal, P. Srinivasan, Predicting performance for gene queries, in: ACM SIGIR, 2005, pp. 1–3.

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

[16] R. Cummins, J. Jose, C. O'Riordan, Improved query performance prediction using standard deviation, in: ACM SIGIR, 2011, pp. 1089–1090.

[17] Z. Zhang, J. Chen, S. Wu, Query performance prediction and classification for information search systems, in: Joint International Conference on Web and Big Data, 2018, pp. 277–285.

[18] H. Zamani, W. B. Croft, J. S. Culpepper, Neural query performance prediction using weak supervision from multiple signals, in: ACM SIGIR, 2018, pp. 105–114.

[19] J. Grivolla, P. Jourlin, R. de Mori, Automatic classification of queries by expected retrieval performance, Actes de SIGIR 5 (2005).

[20] S. Bashir, Combining pre-retrieval query quality predictors using genetic programming, Applied intelligence 40 (2014) 525–535.

[21] A.-G. Chifu, L. Laporte, J. Mothe, M. Z. Ullah, Query performance prediction focused on summarized letor features, in: ACM SIGIR, 2018, p. 1177–1180.

[22] D. Roy, D. Ganguly, M. Mitra, G. J. Jones, Estimating gaussian mixture models in the local neighbourhood of embedded word vectors for query performance prediction, Information Processing & Management 56 (2019) 1026–1045.

[23] S. Datta, D. Ganguly, D. Greene, M. Mitra, Deep-qpp: A pairwise interaction-based deep learning model for supervised qpp, in: ACM WSDM, 2022, pp. 201–209.

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

[25] J. Guo, Y. Fan, Q. Ai, W. B. Croft, A deep relevance matching model for ad-hoc retrieval, in: ACM CIKM, 2016, p. 55–64.

[26] C. Macdonald, R. L. Santos, I. Ounis, B. He, About learning models with multiple query-dependent features, ACM Transactions on Information Systems (TOIS) 31 (2013) 11.

[27] Y. Zhou, W. B. Croft, Query performance prediction in web search environments, in: ACM SIGIR, 2007, p. 543–550.

[28] A. Shtok, O. Kurland, D. Carmel, F. Raiber, G. Markovits, Predicting query performance by query-drift estimation, ACM Trans. Inf. Syst. 30 (2012).

[29] A. Shtok, O. Kurland, D. Carmel, Using statistical decision theory and relevance models for query-performance prediction, in: ACM SIGIR, 2010, p. 259–266.