Deep-QPP: A Pairwise Interaction-based Deep Learning Model for Supervised Query Performance Prediction

Suchana Datta  
University College Dublin, Ireland  
suchana.datta@ucdconnect.ie

Derek Greene  
University College Dublin, Ireland  
derek.greene@ucd.ie

Debasis Ganguly  
University of Glasgow, UK  
debasis.ganguly@glasgow.ac.uk

Mandar Mitra  
Indian Statistical Institute, India  
mandar@isical.ac.in

ABSTRACT
Motivated by the recent success of end-to-end deep neural models for ranking tasks, we present here a supervised end-to-end neural approach for query performance prediction (QPP). In contrast to unsupervised approaches that rely on various statistics of document score distributions, our approach is entirely data-driven. Further, in contrast to weakly supervised approaches, our method also does not rely on the outputs from different QPP estimators. In particular, our model leverages information from the semantic interactions between the terms of a query and those in the top-documents retrieved with it. The architecture of the model comprises multiple layers of 2D convolution filters followed by a feed-forward layer of parameters. Experiments on standard test collections demonstrate that our proposed supervised approach outperforms other state-of-the-art supervised and unsupervised approaches.

CCS CONCEPTS
• Information systems → Query intent; Information retrieval query processing.

KEYWORDS
Supervised Query Performance Prediction, Interaction-based Models, Convolutional Neural Networks

1 INTRODUCTION
The evaluation of information retrieval systems is a challenging problem to solve outside the realm of the Cranfield paradigm [41], i.e., in situations when there are no relevance assessments available, such as those in deployed search systems used by real-life users beyond the laboratory environment. Query performance prediction (QPP) [8, 13, 39, 49, 51], therefore, remains an important and active area of research, because of its usefulness in estimating the quality of a retrieval system on a wide range of queries. The output of a QPP estimator function \( \phi(Q) \) is a likelihood score \( \in \mathbb{R} \), which given a query \( Q \), predicts the retrieval quality of the query. It may therefore, in a sense, be considered to represent how easy (or specific) the query is, because the higher the predicted estimate, the higher is the likelihood that a retrieval model will perform well for the query.

The majority of existing QPP methods relies on devising a suitable heuristic function for predicting the likelihood of how easy a query will be for a retrieval system. Typically, this is estimated by computing the probability of how specific or well-formulated the query is. The specificity measures are computed using either: i) an aggregate of collection statistics over query terms commonly known as pre-retrieval QPP estimators [24, 25]; or by ii) leveraging information from the top-retrieved documents, e.g., assessing the skewness of document similarity scores [39, 51], or measuring the topical differences between the set of top-retrieved documents and the rest of the collection [13].

Supervised deep neural ranking models have recently been shown to improve retrieval effectiveness over their unsupervised statistical counterparts [17, 18, 23, 26, 45]. In contrast to preset similarity functions (e.g. BM25 or LM), these supervised models rely on data-driven parametric learning of similarity functions, usually leveraging an interaction mechanism between the similarities of the embedded representations of constituent words of queries and their returned documents [17, 23, 45].

While the benefits of using supervised approaches have predominantly been established for ranking [5, 12, 18, 26] and recommendation tasks [16, 29, 40, 44], the exploration of supervised approaches for QPP has been limited. The only supervised QPP approach, that we are aware of to the best of our knowledge at the time of writing this paper, is the study [47] which used a combination of features (such as retrieval scores) and word embedded vectors to learn an optimal way of combining a number of different QPP estimates into a single one, thereby outperforming the effectiveness achieved by each individually. A major limitation of [47] is that the training procedure involves weak supervision over a number of estimators to find an optimal combination. In contrast, our proposed method is solely data-driven because it does not rely on other estimators. Moreover, our method is strictly supervised as opposed to the weak supervision employed in [47].
We first describe the working principle of our approach which is to model the interaction between query-document pairs, where the interaction is used to optimize the relative importance of different predictors [47]. Our model makes use of early interactions between a query and its top-retrieved set of documents. We argue that this way of modeling the interaction improves the model's capacity to generalize better as opposed to the late interaction between the content of the queries and the documents [47].

Contributions. In summary, the key contributions of this paper include:

1. An end-to-end supervised QPP model, where instead of learning to optimize the relative importance of different predictors [47], our model learns a comparison function of relative specificity (estimated retrieval quality) between query pairs.

2. Early interactions between query-document pairs, where similar to the deep relevance matching model (DRMM) [23], our model makes use of early interactions between a query and its top-retrieved set of documents. We argue that this way of modeling the interaction improves the model's capacity to generalize better as opposed to the late interaction between the content of the queries and the documents [47].

2 DEEP-QPP MODEL DESCRIPTION

We first describe the working principle of our approach which is based on capturing term-semantics interaction at two levels, first, at the intra-query level of modeling, the interaction between the queries themselves and their top-retrieved documents, and then at the inter-query level, to model their relative specificity measures.

2.1 Representation vs. Interaction

A fundamental difference between a representation-based model and an interaction-based model [23] is illustrated in Figure 1. The former first constructs a representation of each instance from a pair of inputs, and then optimizes this representation so as to maximize the likelihood of predicting a function involving this pair (as seen from the left diagram of Figure 1). In contrast, an interaction-based model first transforms a paired data instance into a single instance via an interaction operator \( \Theta: \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^p \), where \( d \) and \( p \) are the sizes of the raw and the transformed inputs, respectively.

We now discuss the type of interaction suitable for a supervised deep QPP approach. For QPP, the objective function that should be learned from the reference labels is a comparison between a set of top-retrieved documents of a query [39, 51], usually perform better. Motivated by this, we also include information from the top-retrieved documents in the form of early interactions (which we refer to as the intra-query interactions). The parameters of these interactions are then optimized with the help of a late interaction between the queries, which seeks to capture the important characteristic differences of these early interactions towards identifying which query among the pair is easier. An overview of our model is shown in Figure 2.

2.2 Query-Document Interactions

In unsupervised post-retrieval QPP approaches, the interaction between the terms in a query and those of the top-retrieved set takes the form of statically defined functions, which aim to capture how distinct the top-retrieved set is with respect to the collection (e.g., NQC [39] uses the skewness of document retrieval scores, while WIG [51] measures the information gain from the top-retrieved set with respect to the collection). The intra-query interaction shown in Figure 2 involves computing an interaction between the terms of a query and those in its top-retrieved set of documents. This interaction then acts as an input source to learn an optimal specificity function automatically from the data.

Documents to consider for interaction. A common principle that works well as a specificity estimator for post-retrieval QPP approaches, is to measure the distinctiveness between the set of documents towards the top-ranks from the rest of the retrieved set. The standard deviation of the document similarity scores in NQC (i.e., expected difference from the average score), acts as an estimate for the topic distinctiveness of the top set.

Motivated by this insight, in our approach, instead of using only a set of top-k documents, we use information from both the upper and the lower parts of a ranked list. The objective is to capture the differences in the interaction patterns of a set of highly similar (the
upper part of a ranked list) and not-so-highly similar documents (the lower part) as useful cues for QPP.

As notations, we denote the set of documents considered for interaction with a query $Q$ as $R(Q)$, which is comprised of a total of $M = t + b$ documents, including the top-$t$ and the bottom-$b$ ranked ones. The index of the bottom-most document considered for interaction computation is specified by a parameter $N$. This means that the lower part of the ranked list, comprised of $b$ documents are, in fact, those ranked from $N$ to $N - b + 1$. For example, a value of $t = 10$ and $b = 20$ means that $R(Q) = \{D_1, \ldots, D_{100}\} \cup \{D_{101}, \ldots, D_{120}\}$.

In our experiments, we treat $t$ and $b$ as hyper-parameters (see Section 4.4), and restrict $N$ to a value of 100 because it is unlikely that any evidence from documents beyond the top-100 would be useful for the QPP task.

**Interaction between each query term and a document.** We now describe how we compute the query-document interaction matrices for each document $D \in R(Q)$ for a query $Q$. As a first step, we calculate the cosine similarities between the embedded representations of terms – one from the query $Q$ and the other from the document $D_i$. Similar to [23], the distribution of similarities between the $j^{th}$ query term $q_j$ and constituent terms of $D_i$ is then transformed into a vector of fixed length $p$ by the means of 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 $[-1, 1]$).

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

\[
(q_j \odot D_i)^\beta = \sum_{w \in D_i} \mathbf{1}[2(\beta - 1)p - 1 \leq \frac{q_j \cdot w}{|q_j| |w|} < \frac{2\beta}{p} - 1],
\]

where both $q_j \in \mathbb{R}^d$ and $w \in \mathbb{R}^d$, and the interaction vector $q_j \odot D_i^\beta \in \mathbb{R}^p$, and $\mathbf{1}[X] \in \{0, 1\}$ is an indicator variable which takes the value of 1, if a property $X$ is true and 0 otherwise.

**Example 2.1.** If $p = 4$, the interval $[-1, 1]$ is partitioned into the set $\{-1, -0.5, 0, 0.5, 1\}$. For a 3-term document $d$, if the cosine similarities are $0.2, -0.3$ and 0.4 with respect to a query term $q_j$, then $q_j \odot d = \langle 0, 1, 2, 0 \rangle$.

**Collection statistics based relative weighting.** The specificity (i.e., collection statistics, such as idf) of query terms contributes to the effective estimate of QPP scores both in pre-retrieval and post-retrieval approaches. We, therefore, incorporate the idf values of each query term as a factor within the interaction patterns to relatively weight the contributions from the interaction vectors $q_j \odot D_i$. In our proposed approach, we use a generalized version of Equation 1, where we incorporate the idf factor as a part of the interaction vector components, i.e.,

\[
(q_j \odot D_i)^\beta = \log(\frac{N_0}{n(q_j)}) \sum_{w \in D_i} \mathbf{1}[2(\beta - 1)p - 1 \leq \frac{q_j \cdot w}{|q_j| |w|} < \frac{2\beta}{p} - 1],
\]

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.

**Overall interaction between a query and a document.** Each $p$-dimensional interaction vector computed for the $j^{th}$ query term forms the $j^{th}$ row of the overall interaction matrix between the query $Q$ and the $i^{th}$ document $D_i$. The overall interaction matrix, $Q_a \odot D_i^\beta \in \mathbb{R}^{k \times p}$ is thus given by

\[
Q_a \odot D_i^\beta = \{ (q_j \odot D_i)^T \odot (q_k \odot D_i)^T \}^T,
\]

where $k$ is a preset upper limit of the number of terms in a query. A zero-padding is used for the row indices exceeding the number of query terms, i.e., $(q_j \odot D_i) = \{0\}^p, \forall j \geq |Q_a|$. Referring back to Figure 2, each $k \times p$ interaction matrix between a query $Q_a$ and a document $D_i$ corresponds to a colored rectangle (shown in the planes above the queries and documents).

**Interaction between a query and its top-retrieved set.** Finally, each individual document-query interaction matrix, when stacked up one above the other in the order of the document ranks, yields an interaction tensor of order $M \times k \times p$. Formally,

\[
Q_a \odot R(Q_a) = \begin{bmatrix}
Q_a \odot D_1^\beta \\
\vdots \\
Q_a \odot D_M^\beta
\end{bmatrix}
\]

### 2.3 Layered Convolutions for QPP

After constructing the local interactions of a query with its top-retrieved set of documents, i.e. the intra-query interactions, the next step is to extract convolutional features from the $3^{rd}$ order interaction tensor, $Q_a \odot R(Q_a) \in \mathbb{R}^{M \times k \times p}$ between a query $Q_a$ and its top-retrieved set $R(Q_a)$. To this end, 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. Before describing the ways to slicing the tensor into matrices in Section 2.4, we first briefly describe the architecture that we employ to extract useful features from the lower-dimensional slices of the interaction tensor.

**Brief background on 2D convolution.** We do not explain the background of 2D convolution operation [33] in detail. Formally speaking, if $X \in \mathbb{R}^{M \times P}$ represents an input data matrix, and if $W^{(l)} \in \mathbb{R}^{k_l \times k_l}$ ($k_l$ mod 2 = 1, i.e., $k_l$ an odd number) denotes the kernel weight matrix of the $l^{th}$ layer, conveniently represented as $(W^{(l)}_{-k_l/2}, \ldots, 0, \ldots, W^{(l)}_{k_l/2})$, then the outputs of layer-wise convolution, generally speaking, are given by

\[
h_{r,c}^{(l)} = f^{(l)}(\sum_{k \in \mathbb{Z} \setminus \{k_l/2\}} \sum_{j \in \mathbb{Z} \setminus \{k_l/2\}} W^{(l)}_{i,j} h_{r+i,c+j}^{(l-1)}),
\]

for each $l = 1, \ldots, L$ ($L$ being the total number of layers), where $h^{(l-1)} \in \mathbb{R}^{M^{(l-1)} \times P^{(l-1)}}$ is the output obtained from the previous layer of the convolution filter, with $h^{(1)} = X, M^{(1)} = M$ and $P^{(1)} = P$. The function $f^{(l)}$ is an aggregation function that, generally speaking, progressively reduces the size of the convolutional representations, $h^{(l)}$, across layers. Aggregation methods commonly applied in computer vision include the MaxPooling [10, 43] and AvgPooling [37] functions.
Late interactions with convolutional features. A more detailed view of the late interaction across a query pair is shown in Figure 3. Referring to the notation from Equation 5, we employ \( L = 2 \) (i.e. use a total of 2 convolution layers), and use \( k_1 = 5 \) and \( k_2 = 3 \) (i.e. a 5x5 filter for the first layer and a 3x3 for the second one). The aggregate function, \( f^{(l)} \), of each layer \( l \) is set to the MaxPooling operation.

After extracting the convolutional features for each query vs. top-documents interaction tensor (shown as the two cuboids towards the extreme left of Figure 3), we employ the standard practice of merging the convolutional filter outputs of each query into a single vector (shown as the ‘merge’ operation) [7, 42]. Following the merge operation, which now combines abstract features extracted from the local interactions of the two queries into a single vector, we apply a fully connected dense layer. Depending on whether we test the network in a pointwise or pairwise manner, the loss function is set to either the Sigmoid function or a function that seeks to maximize the accuracy of the comparison function between pairs. This may not be desirable because, in standard QPP methods such as WIG and NQC, an individual contribution from each document is set to either the Sigmoid function or a function that seeks to maximize the accuracy of the comparison function between pairs. Section 3 provides more details on the network training process.

### 2.4 Reshaping the Interaction Tensor

There exists a number of different choices for slicing up the interaction tensor of Equation 4 into a set of matrices for the purpose of separately applying 2D convolution on each and then combining the features, shown as the reshaping function \( \pi : R^3 \rightarrow R^2 \) in Figure 3. We now discuss each alternative and examine their pros and cons in the context of the QPP problem.

As our nomenclature, we characterize reshaping functions by whether the information across i) top-retrieved documents are merged together, or across ii) query-terms are merged together. A part of the name thus uses the characters ‘D’ to denote the top-retrieved set, and ‘Q’ to denote query terms. To indicate ‘merging’, we use the letter ‘M’ and to denote its counterpart, we use the letter ‘S’ (separate). For instance, the name MDMQ means that the information from both top-documents and query terms are merged together.

**MDMQ (Merged Documents Merged Query-terms).** This is the most coarse-grained way to reduce the dimensionality of the interaction tensor of order \( 3 \) (Equation 4) by reducing the \( M \times k \times p \) tensor to a flattened vector of dimensionality \( Mkp \), which can still be imagined to be a matrix of dimension \( 1 \times Mkp \) allowing 1D convolutions to be applied. This method extracts abstract features at an aggregate level rather than for individual documents separately. This may not be desirable because, in standard QPP methods such as WIG and NQC, an individual contribution from each document score is responsible for the predicted specificity measure.

**SDMQ (Separate Documents Merged Query-terms).** This corresponds to the most natural way of grouping an interaction tensor, \( Q \oplus R(Q) \), by considering the \( i^{th} \) row for each \( i = 1, \ldots, M \), \( Q \oplus D_i \), as a matrix of dimension \( k \times p \). This method allows the extraction of abstract features from each document separately in relation to the whole query. Thus, it takes into account the compositionality of the query terms, and at the same time avoids mixing information across documents. This conforms to how most unsupervised post-retrieval QPP methods actually work.

**MSQ (Merged Documents Separate Query-terms).** Contrary to grouping the interaction tensor row-wise, for this method we slice out the constituent matrices column-wise. Each matrix is thus of dimension \( M \times p \), and there are a total \( k \) of them, on each of which we apply 2D convolution for feature extraction. This QPP method thus does not take into account the compositionality of the constituent query terms while considering the semantic interactions. Rather it treats the whole set of top-retrieved documents in an aggregated manner, which is also somewhat counter-intuitive because a document at the very top rank should be treated in a different manner from the very bottom one, i.e. the one at \( M^{th} \) rank.
3 Deep-QPP Training

The network in Figure 3 is trained with instances of query pairs with two different objectives – pointwise and pairwise. In the pairwise case, the network directly learns the comparison function, i.e., a binary indicator of the anti-symmetric relation between a query pair. The other hand, the pointwise objective aims to predict a QPP score, instead of the relative order of specificity between a pair. Before describing the objectives, we first provide details on obtaining the data instances and the reference labels.

3.1 Instances and Ground-truth Labels

Given a training set of queries \( Q = \{Q_1, \ldots, Q_m\} \), 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, \( y(Q_a, Q_b) \), of a paired instance is determined by a relative comparison of the retrieval effectiveness obtained by a system with a target metric. The retrieval effectiveness, in turn, is computed with the help of available relevance assessments. Formally speaking, if \( M \) denotes an IR evaluation measure (e.g., average precision or AP), which is a function of \( i \) the known set of relevant documents, \( R(Q) \) for a query \( Q \in Q \), and ii) the set of documents retrieved with a model \( A \) (e.g., LM-Dir [50]), then

\[
y(Q_a, Q_b) = \text{sgn}(M(Q_a; R(Q_a)) - M(Q_b; R(Q_b))), \quad (6)
\]

where \( \text{sgn}(x) = 0 \) if \( x \leq 0 \) or 1 otherwise.

For all our experiments, we used AP@100 and nDCG@20 as the target metric \( M \). As the IR model, \( A \), we employ LM-Dir with the smoothing parameter \( \mu = 1000 \) following QPP literature [39]. We emphasize that the results of our experiments are mostly insensitive to the choice of either the target metric used or the IR model employed.

3.2 Pairwise Objective

For the pairwise objective, the Deep-QPP model is trained to maximize the likelihood of correctly predicting the indicator value of the comparison between a given pair of queries. The purpose here is to learn a data-driven generalization of the comparison function. During the testing phase, the model outputs a predicted value of the comparison between a pair of queries unseen during the training phase. The output layer for the pairwise objective thus constitutes a Sigmoid layer, predicting values of \( y(Q_a, Q_b) \) (see Equation 6) as a function of the network parameters denoted as \( \hat{y}(Q_a, Q_b); \theta \). During the training phase, the parameter updates seek to minimize the standard square loss

\[
\mathcal{L}(Q_a, Q_b) = (y(Q_a, Q_b) - \hat{y}(Q_a, Q_b; \theta))^2 \quad (7)
\]

between the ground-truth and the predicted labels.

3.3 Pointwise Objective

For pointwise testing, as a test input, the network takes a single query \( Q \), as opposed to the pair of queries in the pairwise situation from Section 3.2. Instead of predicting a binary indicator comparison, the network predicts a score \( \hat{y}(Q; \theta) \) that can be used as an estimated measure of specificity of \( Q \). To allow for pointwise testing, the output from the shared layer of parameters goes into a linear activation unit predicting a real-valued score \( \hat{y}(Q; \theta) \), which is a function of one query (rather than a pair), as can be seen from the bottom-right part of the Figure 3. Rather than training the network on a merged representation of a query pair, the loss function includes separate contributions from the two parts of the network corresponding to each query, the objective being to update the parameters for maximizing the comparison agreements between the reference and the predicted scores. Specifically, we minimize the following hinge loss:

\[
\mathcal{L}(Q_a, Q_b) = \max(0, 1 - \text{sgn}(y(Q_a, Q_b) - (\hat{y}(Q_a; \theta) - \hat{y}(Q_b; \theta)))), \quad (8)
\]

4 Experiments

4.1 Datasets and Hyper-parameters

Collections. We experiment with two standard ad-hoc IR test collections, namely the TREC Robust (comprised of news articles) and the ClueWeb09B [11] (comprised of crawled web pages). For the ClueWeb experiments, we used the Waterloo spam scores [4] to remove documents which were detected to be spam with confidence > 70%. We denote this subset as CWb09-B-S70 in Table 1.

Train and test splits. Since our proposed Deep-QPP method is a supervised one, the method first requires a training set of queries to learn the model parameters and then a test set for evaluating the effectiveness of the model. Following the standard convention in the literature, e.g. [39, 47, 49], we employ repeated partitioning (specifically, 30 times) of the set of queries into 50/50 splits and report the average values of the correlation metrics (see Section 4.3) computed over the 30 splits.

A major difference of our setup compared to existing QPP approaches is the use of the training set. While the training set for unsupervised approaches serve the purpose of tuning the hyper-parameters of a model by grid search, in our case, it involves updating the learnable parameters of the neural model by methods such as stochastic gradient descent.

Hyper-parameter Tuning. The most common hyper-parameter for existing unsupervised QPP approaches is the number of top-\( M \)
documents considered for computing the statistics on the document retrieval scores, as in NQC and WIG, or to estimate a relevance feedback model, as in Clarity and UEF (see Section 4.2 for more details). We tune this parameter via grid search on the training partition. As prescribed in [47], the values used in grid search were \{5, 10, 15, 20, 25, 50, 100, 300, 500, 1000\}.

4.2 Baselines

We compare our supervised Deep-QPP approach with a number of standard unsupervised QPP approaches, and also a more recent weak supervision-based neural approach [47]. In our investigation, we do not include QPP methods that leverage external information, such as query variants [6]. Using query variants has been shown to improve the effectiveness of unsupervised QPP estimators and it is also likely that including them in our supervised end-to-end approach will also lead to further improvement in its performance. However, since the main objective of our experiments is to investigate if a deep QPP model can outperform existing ones, we leave the use of external data for future exploration. Moreover, we also do not include the pre-retrieval QPP approaches, such as avg, idf etc., because they have been reported to be outperformed by post-retrieval approaches in a number of existing studies [13, 39, 47, 51].

4.2.1 Unsupervised Approaches. This refers to existing methods that make use of term weight heuristics to measure the specificity estimates of queries. The underlying common principle on which all these methods rely is the assumption that, if the set of top-ranked documents retrieved for a query is substantially different from the rest of the collection, then the query is likely to be indicative of unambiguous information need. This makes it a potentially good candidate for achieving effective retrieval results. These methods mainly differ in the way in which they calculate the similarity of the top-retrieved set of documents from the rest of the collection.

Clarity [13]. This method estimates a relevance model (RLM) [28] distribution of term weights from a set of top-ranked documents, and then computes its KL divergence with the collection model - the higher the KL divergence (a distance measure) the higher is the query specificity.

WIG [51]. 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 the overall set of top-documents.

NQC [39]. 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. NQC thus makes use of not only the relative gain of a document score from the collection (similar to WIG) but also the gain in a document’s score with respect to the average score.

UEF [38]. The UEF method assumes that information from some top-retrieved 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, which in turn suggests that a smaller confidence should be associated with the QPP estimate of such a query.

4.2.2 Supervised Approaches. Our choice of supervised baselines is guided by two objectives - first, to show that (strong) supervision using the ground-truth of relative query performance is better than the existing approach of weak supervision on QPP estimation functions [47], and second, to demonstrate that a mixture of both early and late interactions (i.e., a hybrid of both content and interaction-focused approaches) is better than purely content-based ones (see Figures 1 and 2).

Weakly Supervised Neural QPP (WS-NeurQPP) [47]. The main difference between WS-NeurQPP and Deep-QPP lies in the source of information used and also the objective of the neural end-to-end models. WS-NeurQPP uses weak supervision to approximate the scores of individual QPP estimators so as to learn an optimal combination. As inputs, it uses the retrieval scores, along with the word embedded vectors. However, in contrast to our approach, it does not use interactions between terms and is hence a purely representation-based approach.

Siamese Network (SN). This approach is an ablation of the Deep-QPP model (Figure 3). Here instead of feeding in the interaction tensors between a query and its top-retrieved documents, we simply input the dense vector representations of queries in pairs. We experiment with two different types of dense vector inputs - one where we used pre-trained RoBERTa vectors [30] obtained using the HuggingFace library [2], and the other, where we used the sum of the Skipgram [31] word embedded vectors (trained on the respective target collections) of constituent terms as the dense representation of a query for input. We name these two ablations as SN-BERT and SN-SG, respectively.

No Intra-Query Interaction. As another ablation of Deep-QPP, we only use the interaction between the terms of the query pairs themselves. The interaction tensor between a pair of queries is a 2nd order tensor, i.e., a $k \times p$ matrix. This is a purely interaction-based method, and in principle, is similar to DRMM [23], with the added layer of 2D convolutions. We denote this baseline as DRMM.

4.3 Experiment Settings

Implementation. We used the Java API of Lucene 8.8 [1] for indexing and retrieval; also to implement the existing unsupervised QPP baselines (e.g., for calculating the document and collection statistics). The supervised baseline - WS-NeurQPP, and our proposed method - Deep-QPP, were both implemented in Keras [3]. The code for our proposed method is available for research purposes1.

Metrics. Recall from Section 3 that the Deep-QPP model can be trained using either the pairwise and the pointwise objectives. The pointwise test use-case is the standard practice in existing QPP

1https://github.com/suchanadatta/DeepQPP.git


Table 2: A comparison of the QPP effectiveness between Deep-QPP, and a set of unsupervised and supervised baselines (shown in the 1st and the 2nd groups, respectively). The average accuracy and the correlation values (see Section 4.3) of Deep-QPP over the best performing baseline - WS-NeurQPP, are statistically significant (t-test with over 97% confidence).

| Methods                  | Metric : AP@100 | Metric : nDCG@20 |
|--------------------------|------------------|------------------|
|                          | TREC-Robust      | ClueWeb09B       | TREC-Robust      | ClueWeb09B       |
|                          | Pairwise         | Pointwise        | Pairwise         | Pointwise        |
|                          | Accuracy P-𝜌 K-𝜌 | Accuracy P-𝜌 K-𝜌 | Accuracy P-𝜌 K-𝜌 | Accuracy P-𝜌 K-𝜌 |
| Clarity [13]             | 0.6251 0.4863 0.3140 | 0.6120 0.1911 0.0641 | 0.6118 0.3529 0.2462 | 0.6101 0.0923 0.0714 |
| NQC [39]                 | 0.6720 0.5269 0.4041 | 0.7030 0.2654 0.1518 | 0.6689 0.4261 0.3017 | 0.6916 0.3105 0.1987 |
| WIG [51]                 | 0.6613 0.5440 0.4279 | 0.6829 0.2492 0.1920 | 0.6629 0.3915 0.2407 | 0.6710 0.2780 0.1823 |
| UEF [38]                 | 0.6941 0.5523 0.4154 | 0.7217 0.3162 0.1959 | 0.6792 0.5029 0.3510 | 0.6925 0.3320 0.1854 |
| SN-BERT                  | 0.6613 0.5208 0.4169 | 0.6902 0.2317 0.1441 | 0.6529 0.5023 0.3624 | 0.6724 0.2241 0.1334 |
| SN-SG                    | 0.6349 0.5112 0.3987 | 0.6273 0.2110 0.1154 | 0.6147 0.4736 0.3561 | 0.6231 0.2049 0.1283 |
| DRMM                     | 0.5871 0.4730 0.3710 | 0.6023 0.2014 0.1141 | 0.5629 0.4038 0.3119 | 0.6004 0.1927 0.1201 |
| WS-NeurQPP [47]          | 0.8123 0.7215 0.5090 | 0.7727 0.5192 0.2828 | 0.7973 0.5913 0.4126 | 0.7614 0.3928 0.2337 |
| Deep-QPP (MDMQ)          | 0.7857 0.6988 0.4981 | 0.7414 0.4636 0.2495 | 0.7632 0.5649 0.3619 | 0.7189 0.3509 0.2185 |
| Deep-QPP (SDSQ)          | 0.7210 0.6303 0.4018 | 0.6844 0.4208 0.2401 | 0.7284 0.5112 0.3065 | 0.6753 0.3124 0.2014 |
| Deep-QPP (MDSQ)          | 0.8006 0.7203 0.4989 | 0.7426 0.4840 0.2575 | 0.7824 0.5601 0.3245 | 0.7037 0.3518 0.2100 |
| Deep-QPP (SDMQ)          | 0.8420 0.7404 0.5454 | 0.8045 0.5532 0.3130 | 0.8371 0.6315 0.4614 | 0.7903 0.4431 0.2554 |

Deep-QPP hyper-parameters. For the Deep-QPP method (and also for the semantic analyzer component of the weakly supervised baseline WS-NeurQPP), we use skip-gram word vectors of dimension 300 trained on the respective document collections with a window size of 10 and 25 negative samples. Another hyper-parameter in Deep-QPP is the number of intervals (bins) \( p \) used to compute the interactions in Equation 2. In Table 2, we report results with \( p = 30 \) (as per the settings of the DRMM paper [23]), and later investigate the effect of varying this parameter on the effectiveness of Deep-QPP (results in Figure 6).

We observed that, after a number of initial experiments, excluding the idf of terms in the interaction tensors always produced worse results than when including them. Therefore, in all our experiments with Deep-QPP, we use the idf-weighted interactions (Equation 2), and do not report the results obtained with Equation 1 for brevity. Another hyper-parameter that we use in the Deep-QPP model to avoid over-fitting is the dropout probability, which we set to 0.2 as per the initial trends in our experimental findings.

4.4 Results

Table 2 presents the QPP results for all the methods investigated. Firstly, we observe that the existing supervised approach for QPP, WS-NeurQPP, outperforms the unsupervised approaches (NQC, WIG, and UEF), which conforms to the observations reported in [47]. Secondly, we observe that the ablation baselines of Deep-QPP involving a purely representation-based approach (SN-BERT and SN-SG), or a purely interaction-based one (DRMM), perform worse than Deep-QPP. This is mainly because these baselines lack the additional source of information – interactions of queries with the top-retrieved set of documents, which Deep-QPP is able to leverage from. This observation also reflects the fact that post-retrieval QPP approaches, with the additional information from top-documents, typically outperform pre-retrieval ones [39].

Third and most importantly, we observe that Deep-QPP outperforms WS-NeurQPP, which confirms the hypothesis that explicitly learning the relative specificity of query pairs with an end-to-end (strongly) supervised model is better able to generalize than a weakly supervised approach which learns an optimal combination of statistical predictors.

Another observation is that the SDMQ version of the reshaping function \( \pi : \mathbb{R}^3 \mapsto \mathbb{R}^2 \) (see Section 2.4 and Figure 3) turns out to be the most effective, as we might expect. This also conforms to the way in which unsupervised QPP approaches generally work, i.e., by first making use of the information from each top-retrieved document (e.g., its score in NQC and WIG) and then computing an aggregate function over them (e.g., their variance in NQC, and relative gains in WIG).

To further compare Deep-QPP to WS-NeurQPP, we report the training-time efficiency of both approaches in Figure 4. Due to...
We have already discussed a number of existing QPP methods as a part of the description of the baselines in Section 4.2. We now outline additional QPP work, and also cover some recent work on applications of end-to-end learning in IR. Kurland et al. [27] showed that the QPP task is equivalent to ranking clusters of similar documents by their relevance with respect to a query. Zedel et al. [49] made use of alternative expressions of information needs, such as variants of a given query, to improve QPP effectiveness. The study [21] reported that a spatial analysis of vector representations of top-retrieved documents provide useful cues for improving QPP effectiveness – a hypothesis that our data-driven model also includes, through the convolutions over the interaction matrices. Other standard deviation-based approaches, somewhat similar to NJC, have also been reported to work well for the QPP task [14, 15]. Apart from the weakly supervised neural approach of WS-NeurQPP [47], a QPP unsupervised approach that uses cluster hypothesis of word vectors in an embedded space was proposed in [35].

Recent studies have reported a close association between the findings of learning to rank (LTR) and QPP studies. It was reported that the set of features that are useful for LTR also proves beneficial for QPP [9, 19]. Moreover, the mechanism of two levels of interaction (both between queries and documents, and across queries) has also been reported to be useful for LTR [32].

In addition to DRMM [23], other work proposing end-to-end LTR approaches include [45, 48]. The ColBERT model was recently proposed in [26], which is a fine-tuned BERT model [20] using pairwise ranking loss. As a precursor to end-to-end supervised approaches, unsupervised approaches have addressed term semantics by using dense word vectors, including [22, 34, 36] which used skip-gram vectors and the work of [46] which employed BERT.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed Deep-QPP, a data-driven end-to-end neural framework for the task of query performance prediction in ad-hoc retrieval. Rather than relying on statistical term weighting heuristics or employing a weakly-supervised model on these heuristics, our method directly learns from the data, where the input consists of a set of queries, along with their top-retrieved sets of documents. The ground-truth for training is comprised of the true query performance indicators (e.g., measured with AP). Our experiments, conducted on standard news and Web collections, demonstrated that a data-driven approach trained on query pairs with known QPP indications (e.g., AP values) is able to effectively generalize this comparison function for unseen query pairs. The improvement percentages obtained for Web queries are in fact higher which suggest that, in future we could potentially use pseudo-relevance information in the context of query logs, such as clicks and dwell times, to train QPP models at a large scale.

Acknowledgement. The first and the third authors were partially supported by the Science Foundation Ireland (SFI) grant number SFI/12/RC/2289_P2.
