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

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 the query and those of the top-retrieved documents. The architecture of the model comprises multiple layers of 2D convolution filters followed by a feed-forward layer of parameters. Experiments on standard test collectionsdemonstrate that our proposed supervised approach outperforms other state-of-the-art supervised and unsupervised approaches.

KEYWORDS

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

ACM Reference Format:

. 2022. Deep-QPP: A Pairwise Interaction-based Deep Learning Model for Supervised Query Performance Prediction. In *WSDM '22: ACM International Conference on Web Search and Data Mining, Phoenix, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The evaluation of information retrieval systems is a challenging problem to solve outside the realm of the Cranfield paradigm [40], 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, 12, 38, 48, 50], 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 the likelihood that a retrieval model will perform well for the query. An effective QPP estimate is thus expected to exhibit a high correlation with retrieval effectiveness measures.

WSDM '22, Feb 21–25, 2022, Phoenix, AZ, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The majority of existing QPP methods rely 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, based on the assumption that an IR model is expected to perform well (i.e. retrieve a large number of relevant documents towards top ranks) for well-formulated queries. 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 [38, 50], or measuring the topical differences between the set of top-retrieved documents and the rest of the collection [12].

Supervised deep neural ranking models have recently been shown to improve retrieval effectiveness over their unsupervised statistical counterparts [16, 17, 20, 23, 27, 44]. 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 retrieved documents [16, 23, 44].

While the benefits of using supervised approaches have predominantly been established for ranking [5, 11, 17, 27, 47] and recommendation tasks [15, 21, 26, 30, 39, 43], 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 [46]. The authors of [46] 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 [46] 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 [46].

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 [46], 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 constituting the input improves its capacity to generalize better as opposed to the late interaction between the content of the queries and the documents. [46].

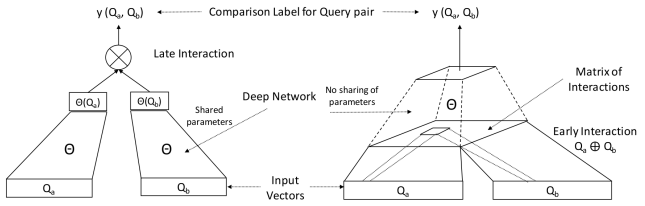

Figure 1: While representation-based models rely on *late interaction* involving shared parameters (left), interaction-based models, on the other hand, make use of *early interactions* transforming paired instances into a single input.

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 $\oplus : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^p$, 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. Referring back to the general workflow of the models for QPP the objective function that should be learned from the reference labels is a comparison between a pair of queries, Q_a and Q_b. More concretely, this comparison is an indicator of the relative difficulty between the queries, i.e., whether Q_a is more difficult than Q_b or vice versa.

While pre-retrieval QPP approaches only rely on the information from a query itself (e.g., aggregate collection statistics for its terms [24, 25]), it has been shown that post-retrieval approaches, which make use of additional information from the top-retrieved documents of a query [38, 50], 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 given 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 [38] uses the skewness of document retrieval scores, while

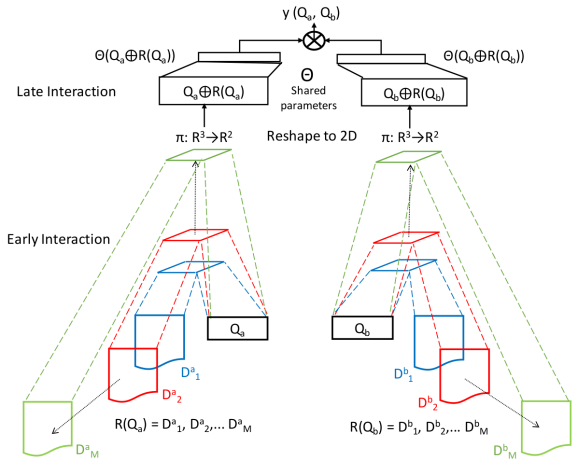

Figure 2: Unlike an entirely representation-based or interaction-based model (Figure 1), our model combines the benefits of both early and late interactions, to address: a) the interaction of the terms in the top-retrieved documents of a query with the constituent terms of the query itself; b) the characteristic pattern of these interactions to estimate the comparison function $y(Q_a, Q_b)$ between a pair of queries. Each individual query-document interaction is shown with a different color.

WIG [50] 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 $R(Q) = \{D_1, \ldots, D_{10}\} \cup \{D_{81}, \ldots, D_{100}\}$.

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_a and the other from the document D_i^a. Similar to [23], the distribution of similarities between the j^{th} query term q_j and constituent terms of D_i^a 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 β^{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 β^{th} partition of $[-1, 1)$, i.e.,

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

where both $\mathbf{q}_j \in \mathbb{R}^d$ and $\mathbf{w} \in \mathbb{R}^d$, and the interaction vector $q_i \oplus D_i^a \in \mathbb{R}^p$, and $\mathbb{I}[X] \in \{0, 1\}$ 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 $\{[-1, -0.5), [-0.5, 0), [0, 0.5), [0.5, 1)\}$. 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, then $q \oplus d = (0, 1, 2, 0)$.

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 weigh the contributions from the interaction vectors $q_j \oplus D_i^a$. 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.,

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

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_a and the i^{th} document D_i^a. The overall interaction matrix, $Q_a \oplus D_i^a \in \mathbb{R}^{k \times b}$ is thus given by

$$Q_a \oplus D_i^a = [(q_1 \oplus D_i^a)^T, \ldots, (q_k \oplus D_i^a)^T]^T, \quad (3)$$

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., $(q_j \oplus D_i^a) = \{0\}^b$, $\forall j > |Q_a|$. Referring back to Figure 2, each $k \times p$ interaction matrix between a query Q_a and a document D_i^a 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,

$$Q_a \oplus R(Q_a) = \begin{bmatrix} Q_a \oplus D_1^a \\ \cdots \\ \cdots \\ Q_a \oplus D_M^a \end{bmatrix} \quad (4)$$

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, $Q_a \oplus R(Q_a) \in \mathbb{R}^{M \times k \times b}$ 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 in Section 2.4 the ways to slicing the tensor into matrices, we first 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 start with a brief background on the 2D convolution operation itself. This operation is inspired from how the visual cortex in the human brain processes information, namely by responding to changes in a local spatial neighborhood of a 2D signal [33]. To emulate the local receptive fields of visual cortex, a convolutional neural network (CNN) performs a discrete convolution operation, which involves shifting a *convolution kernel* (i.e., a matrix of shared learnable weights) on an input 2D data. The output or activation value of each neuron is calculated by multiplying the local input with the weights. Usually these convolution operations are conducted in a layer-wise manner, with the output from the previous layer being fed as input data to the next layer, where each layer potentially has a different kernel size and its own set of parameters.

Formally speaking, if $\mathbf{X} \in \mathbb{R}^{M \times P}$ represents an input data matrix, and if $\mathbf{W}^{(l)} \in \mathbb{R}^{k_l \times k_l}$ ($k_l \mod 2 = 1$, i.e., k_l an odd number) denotes the kernel weight matrix of the l^{th} layer, conveniently represented as $(W^{(l)}_{-\lfloor k/2 \rfloor}, \ldots, 0, \ldots, W^{(l)}_{\lfloor k/2 \rfloor})$, then the outputs of layer-wise convolution, generally speaking, are given by

$$\mathbf{h}^{(l)}_{r,c} = f^{(l)}\Big(\sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \mathbf{W}^{(l)}_{i,j} \mathbf{h}^{(l-1)}_{r+i,c+j}\Big), \quad (5)$$

for each $l = 1, \ldots, L$ (L being the total number of layers), where $\mathbf{h}^{(l-1)} \in \mathbb{R}^{M^{(k-1)} \times P^{(k-1)}}$ 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, $\mathbf{h}^{(l)}$, across layers. Aggregation methods commonly applied in computer vision include the MaxPooling [9, 42] and AvgPooling [36] 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

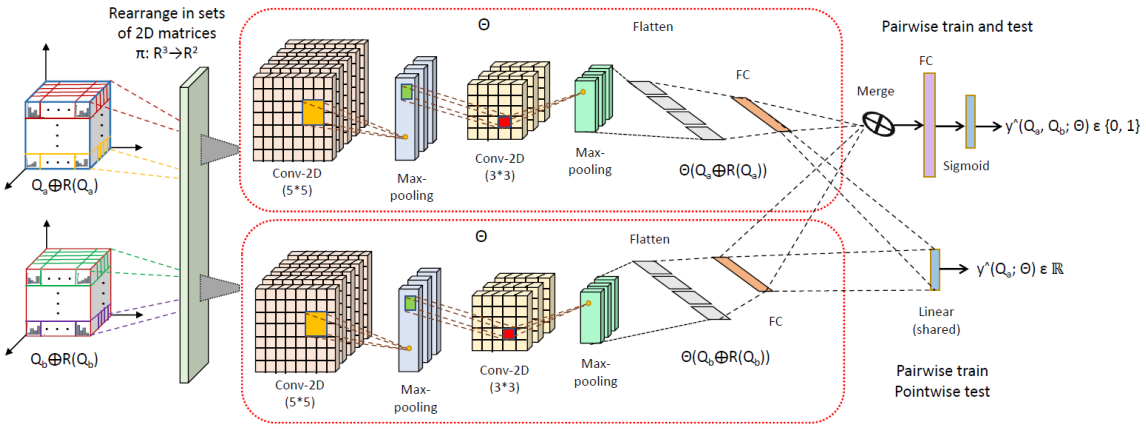

Figure 3: Our proposed end-to-end QPP model comprising a Siamese network of shared parameters of layered convolutional feature extraction, followed by either i) merge (concatenation) and a fully connected (FC) layer with a Sigmoid **loss for pairwise testing (Equation 7) yielding a binary comparison indicator between a pair, or ii) a linear activation layer with pairwise hinge loss for pointwise testing yielding a score for a given query (Equation 8). Since the interaction for MDMQ and SDSQ are matrices with a single row only, the two layers of convolution filter sizes for these approaches are** 1×5 **and** 1×3 **(see Section 2.4).**

$L = 2$ (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, 41]. 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. Section 3 provides more details on the network training process.

2.4 Reshaping the Interaction Tensor

There exist 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 $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ 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.

MDSQ (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.

SDSQ (Separate Documents Separate Query-terms). This is the most fine-grained approach, which considers every interaction vector between the j^{th} query term and i^{th} document (see Equation

2 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 Mk such vectors. As with the MDMQ approach, we apply 1D convolution on these vectors.

A point to note is that, although Figure 3 shows the convolution filters as 5×5 and 3×3, for MDMQ and SDSQ approaches these filters are of size 1×5 and 1×3 respectively.

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. On 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 $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$, 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, $y(Q_i, Q_j)$, 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 $\mathcal{M}$ denotes an IR evaluation measure (e.g., average precision or AP), which is a function of i), the known set of relevant documents - $\mathcal{R}(Q)$ for a query $Q \in \mathcal{Q}$, and ii) the set of documents retrieved with a model $\mathcal{A}$ (e.g., LM-Dir [49]), then

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

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 $\mathcal{M}$. As the IR model, $\mathcal{A}$, we employ LM-Dir with the smoothing parameter $\mu = 1000$. 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.

Coll	#Docs	Topic Set	$\lvert\mathcal{Q}\rvert$	Avg. Q_len	Avg. #Rel
Disks 4 & 5	528,155	TREC-Rb	249	2.68	71.21
CWeb09B-S70	29,038,220	TREC-Web	200	2.42	16.03

Table 1: Dataset Characteristics (the suffix 'S70' indicates that documents detected as spams with confidence scores higher than 70% were removed from the collection).

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:

$$\mathcal{L}(Q_a, Q_b) = \max(0, 1 - \text{sgn}(y(Q_a, Q_b) \cdot (\hat{y}(Q_a; \Theta) - \hat{y}(Q_b; \Theta)))). \quad (8)$$

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 [10] (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 CWeb09B-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. [38, 46, 48], 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 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 [46], 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 [46]. 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 may 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 [12, 38, 46, 50]. We now enlist the baseline methods investigated.

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-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 [12]. This method estimates a relevance model (RLM) [29] 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 [50]. 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 [38]. 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 [37]. The UEF method assumes that information from some top-retrieved sets 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, 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 [46], 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) [46]. 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. As prescribed in [46], we make use of the three individual components of the network, namely the a) retrieval scores analyzer, b) the term distribution analyzer, and c) the semantic analyzer, respectively taking as inputs the document similarity scores, the term-document matrix of top-documents, and the dense vectors of words within the top-documents.

Siamese Network (SN). This approach is an ablation of the Deep-QPP model (Figure 3), where instead of leveraging information in the form of early interactions between a query and its top-retrieved set of documents, we simply use the late interaction mechanism with the help of a Siamese network with shared parameters, the architecture of which is identical to Deep-QPP. 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 [31] obtained using the HuggingFace library [2], and the other, where we used the sum of the Skipgram [32] 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 2^{nd} 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. We also used the Lucene indexing APIs to implement the existing unsupervised QPP baselines (e.g., for calculating the document and collection statistics). The supervised

Methods	Metric : AP@100						Metric : nDCG@20					
	TREC-Robust			ClueWeb09B			TREC-Robust			ClueWeb09B		
	Pairwise	Pointwise		Pairwise	Pointwise		Pairwise	Pointwise		Pairwise	Pointwise	
	Accuracy	P-ρ	K-τ	Accuracy	P-ρ	K-τ	Accuracy	P-ρ	K-τ	Accuracy	P-ρ	K-τ
Clarity [12]	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 [38]	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 [50]	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 [37]	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[46]	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**	**0.5434**	**0.8045**	**0.5532**	**0.3130**	**0.8371**	**0.6315**	**0.4614**	**0.7903**	**0.4431**	**0.2554**

Table 2: A comparison of the QPP effectiveness between Deep-QPP, and a set of unsupervised and supervised baselines (shown in the 1^{st} and the 2^{nd} 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).

baseline - WS-NeurQPP, and our proposed method - Deep-QPP, were both implemented in Keras [3]. The code for our proposed method and the baselines is available for research purposes[1].

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 studies, where given a query, a QPP model predicts a score indicative of the retrieval effectiveness. For this use-case, we evaluate the effectiveness of the QPP methods with standard metrics used in the literature: a) Pearson's-ρ correlation between the AP values of the queries in the test-set and the predicted QPP scores; b) a ranking correlation measure, specifically Kendall's τ between the ground-truth ordering (increasing AP values) of the test-set queries and the ordering induced by the predicted QPP scores.

In pairwise testing, the network is presented with pairs of queries from the test set, for which it then predicts binary indications of the relative order of queries within the pairs. As a QPP effectiveness measure, we report the average accuracy of these predictions, i.e. whether a predicted relation as given by the Sigmoid output from Deep-QPP, $\hat{y}(Q_a, Q_b; \Theta)$, matches the ground-truth that $AP(Q_a) < AP(Q_b)$. Since $\hat{y}(Q_a, Q_b; \Theta) \in [0, 1]$, we *binarize* this value to $\{0, 1\}$ with the threshold of 0.5, thus indicating a prediction of whether Q_a is a more difficult query than Q_b or vice versa.

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 [46]. *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 [38].

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

[1]Github link not provided in this review version of the draft for anonymity.

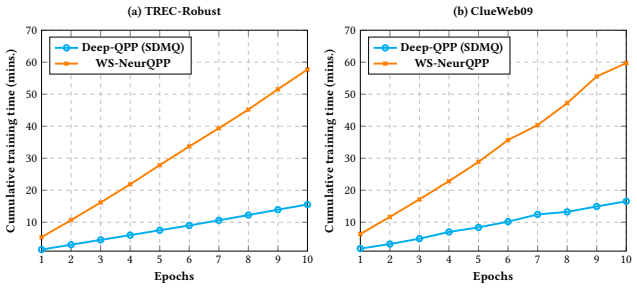

Figure 4: Deep-QPP, in addition to being more effective than WS-NeurQPP, also outperforms WS-NeurQPP in terms of training time because of a much smaller number of parameters (1.9M vs. 4.7M).

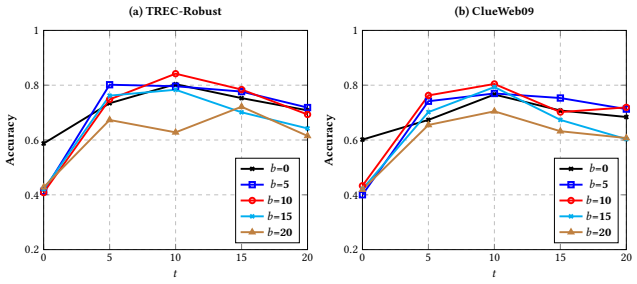

Figure 5: Sensitivity of Deep-QPP on the number of top (t) and bottom (b) documents to include for interaction computation (see Section 2.2) on QPP effectiveness. The limiting case of $(t, b) = (0, 0)$ corresponds to the situation when we simply use the interaction between query terms themselves (i.e. the DRMM baseline).

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 a much larger number of trainable parameters and larger input dimensionality (dense word vectors instead of interactions between the dense vectors), WS-NeurQPP turns out to be taking a much larger time to execute than Deep-QPP. The total number of trainable parameters of WS-NeurQPP is 4.7M which is about 2.5X the number of parameters in Deep-QPP (1.9M).

Hyper-parameter Sensitivity of Deep-QPP. Figure 5 shows that using the top-10 and the bottom-10 documents for the interaction computation (Section 2.2) yields the best results, which shows that neither a too small nor too large a number of documents should be used as inputs for learning the QPP comparison function.

Figure 6 shows the effects of different bin-sizes, p (of Equation 2), used to compute the interactions between queries and the documents retrieved at top and bottom ranks. A value of 30 turned out to be optimal, which is similar to the reported optimal value of the bin-size for interaction computation in the LTR task [23].

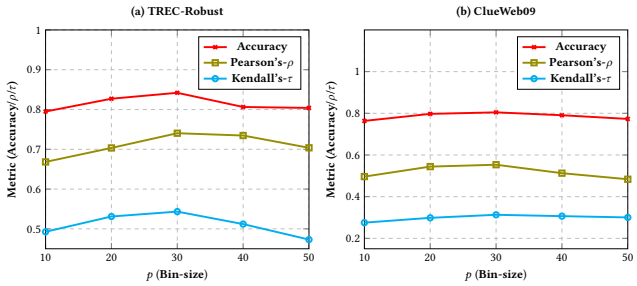

Figure 6: Sensitivity of Deep-QPP w.r.t. the bin-size, p.

5 RELATED WORK

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. [28] showed that the QPP task is equivalent to ranking clusters of similar documents by their relevance with respect to a query. Zendel et. al. [48] made use of alternative expressions of information needs, such as variants of a given query, to improve QPP effectiveness. The study [19] 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 NQC, have also been reported to work well for the QPP task [13, 14]. Apart from the weakly supervised neural approach of WS-NeurQPP [46], a QPP unsupervised approach that uses cluster hypothesis of word vectors in an embedded space was proposed in [34].

In addition to DRMM [23], other work proposing end-to-end LTR approaches include [44, 47]. The ColBERT model was recently proposed in [27], which is a fine-tuned BERT model [18] 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, 35] which used skip-gram vectors and the work of [45] 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 those 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.

REFERENCES

[1] 2021. Apache Lucene. https://lucene.apache.org/. Accessed: 2021-08-13.

[2] 2021. Huggingface Transformers. https://huggingface.co/transformers/. Accessed: 2021-08-13.

[3] 2021. Keras. https://keras.io/. Accessed: 2021-08-13.

[4] 2021. Waterloo Spam Rankings for the ClueWeb09 Dataset. https://plg.uwaterloo.ca/~gvcormac/clueweb09spam/. Accessed: 2021-08-13.

[5] Nima Asadi, Donald Metzler, Tamer Elsayed, and Jimmy Lin. 2011. Pseudo Test Collections for Learning Web Search Ranking Functions. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. 1073–1082.

[6] Olga Butman, Anna Shtok, Oren Kurland, and David Carmel. 2013. Query-Performance Prediction Using Minimal Relevance Feedback. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval (ICTIR '13)*. 14–21.

[7] Adam Byerly, Tatiana Kalganova, and Ian Dear. 2020. A Branching and Merging Convolutional Network with Homogeneous Filter Capsules. *CoRR* abs/2001.09136 (2020).

[8] David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. 2006. What Makes a Query Difficult?. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. 390–397.

[9] Vincent Christlein, Lukas Spranger, Mathias Seuret, Anguelos Nicolaou, Pavel Král, and Andreas Maier. 2019. Deep Generalized Max Pooling. *2019 International Conference on Document Analysis and Recognition (ICDAR)* (2019), 1090–1096.

[10] C. Clarke, Nick Craswell, I. Soboroff, and G. Cormack. 2010. Overview of the TREC 2010 Web Track. In *TREC*.

[11] Daniel Cohen, John Foley, Hamed Zamani, James Allan, and W. Bruce Croft. 2018. Universal Approximation Functions for Fast Learning to Rank: Replacing Expensive Regression Forests with Simple Feed-Forward Networks. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '18)*. 1017–1020.

[12] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting Query Performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*. 299–306.

[13] Ronan Cummins. 2014. Document Score Distribution Models for Query Performance Inference and Prediction. *ACM Trans. Inf. Syst.* 32, 1, Article 2 (2014).

[14] Ronan Cummins, Joemon Jose, and Colm O'Riordan. 2011. Improved Query Performance Prediction Using Standard Deviation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. 1089–1090.

[15] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. *CoRR* abs/1907.06902.

[16] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-Hoc Search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. 126–134.

[17] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. 65–74.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805.

[19] Fernando Diaz. 2007. Performance Prediction Using Spatial Autocorrelation. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 583–590.

[20] Yongyu Jiang et al. 2020. *A Quantum Interference Inspired Neural Matching Model for Ad-Hoc Retrieval.* 19–28.

[21] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference (WWW '19)*. 417–426.

[22] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word Embedding Based Generalized Language Model for Information Retrieval *(SIGIR '15)*. 795–798.

[23] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-Hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. 55–64.

[24] Claudia Hauff. 2010. Predicting the Effectiveness of Queries and Retrieval Systems. *SIGIR Forum* 44, 1 (Aug. 2010), 88.

[25] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. 2008. A Survey of Pre-Retrieval Query Performance Predictors. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*. 1419–1420.

[26] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Trans. on Knowl. and Data Eng.* 30, 12 (2018), 2354–2366.

[27] Omar Khattab and Matei Zaharia. 2020. *ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT.* 39–48.

[28] Oren Kurland, Fiana Raiber, and Anna Shtok. 2012. Query-Performance Prediction and Cluster Ranking: Two Sides of the Same Coin. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. 2459–2462.

[29] Victor Lavrenko and W. Bruce Croft. 2001. Relevance Based Language Models. In *Proc. of SIGIR '01*. 120–127.

[30] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-Based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[31] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. 3111–3119.

[33] A. Rodríguez-Sánchez, M. Fallah, and A. Leonardis. 2015. Editorial: Hierarchical Object Representations in the Visual Cortex and Computer Vision. *Frontiers in Computational Neuroscience* 9 (2015).

[34] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and G. Jones. 2019. Estimating Gaussian mixture models in the local neighbourhood of embedded word vectors for query performance prediction. *Inf. Process. Manag.* 56 (2019), 1026–1045.

[35] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J.F. Jones. 2016. Word Vector Compositionality Based Relevance Feedback Using Kernel Density Estimation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. 1281–1290.

[36] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. 101–110.

[37] Anna Shtok, Oren Kurland, and David Carmel. 2010. Using Statistical Decision Theory and Relevance Models for Query-Performance Prediction. In *In Proc. of SIGIR'10 (SIGIR '10)*. 259–266.

[38] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting Query Performance by Query-Drift Estimation. *ACM Trans. Inf. Syst.* 30, 2, Article 11 (2012), 35 pages.

[39] Elena Smirnova and Flavian Vasile. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems (DLRS 2017)*. 2–9.

[40] Ellen M. Voorhees. 2001. The Philosophy of Information Retrieval Evaluation. In *Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems (CLEF '01)*. 355–370.

[41] Kunfu Wang, Pengyi Zhang, and Jian Su. 2020. A Text Classification Method Based on the Merge-LSTM-CNN Model. 1646 (2020), 012110.

[42] Haibing Wu and Xiaodong Gu. 2015. Max-Pooling Dropout for Regularization of Convolutional Neural Networks. In *Neural Information Processing*, Sabri Arik, Tingwen Huang, Weng Kin Lai, and Qingshan Liu (Eds.). 46–54.

[43] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. *CoRR* abs/1904.10322.

[44] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. 55–64.

[45] Zeynep Akkalyoncu Yilmaz, S. Wang, W. Yang, Haotian Zhang, and Jimmy J. Lin. 2019. Applying BERT to Document Retrieval with Birch. In *Proc. 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 19–24.

[46] Hamed Zamani, W Bruce Croft, and J Shane Culpepper. 2018. Neural Query Performance Prediction Using Weak Supervision from Multiple Signals. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '18)*. 105–114.

[47] Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. 2018. Neural Ranking Models with Multiple Document Fields. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

[48] Oleg Zendel, Anna Shtok, Fiana Raiber, Oren Kurland, and J. Shane Culpepper. 2019. Information Needs, Queries, and Query Performance Prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 395–404.

[49] Chengxiang Zhai and John Lafferty. 2001. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. 334–342.

[50] Yun Zhou and W. Bruce Croft. 2007. Query Performance Prediction in Web Search Environments. In *Proc. 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 543–550.