Information Sciences 639 (2023) 119015

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Learning to rank and predict: Multi-task learning for ad hoc retrieval and query performance prediction

Maryam Khodabakhsh [a,*], Ebrahim Bagheri [b]

[a] Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran
[b] Laboratory for Systems, Software and Semantics (LS^3), Toronto Metropolitan University, Canada

ARTICLE INFO

Keywords:
Multi-task learning
Query performance prediction
Ad hoc retrieval
BERT

ABSTRACT

The ad hoc retrieval task aims at ranking relevant documents to a user query such that the most relevant documents are ranked higher compared to less relevant ones. Given the performance of the ad hoc retrieval task can vary across a range of queries, researchers have extensively explored the interrelated task of query performance prediction, which aims at estimating the quality of the search results for a user query without having access to relevance judgments. Traditionally and to-date, the two tasks have been explored as separate tasks where ad hoc retrieval and query performance prediction have been performed in isolation. In this paper, we propose to learn joint tasks that would perform ad hoc retrieval and at the same time predict the quality of the produced rankings. More specifically, we propose a multi-task learning approach, called Multi-task Query Performance Prediction Framework (M-QPPF), which learns document ranking and query performance prediction tasks simultaneously. In M-QPPF, we adopt a shared BERT layer, which is fine-tuned to learn representations for query-document pairs in the embedding space such that the representations effectively encode the cross-interaction between the query and documents. In addition, we include additional yet separate layers to capture task-specific characteristics. We perform comprehensive experiments against state-of-the-art methods in the query performance prediction and ranking tasks over large-scale datasets including the MS MARCO and TREC DL datasets. The improvements measure up to 18.8% on MS MARCO dataset with a Pearson correlation of 0.604, which is superior to the performance of any state-of-the-art baseline in the query performance prediction task.

1. Introduction

With the growing influence of search engines on users' information-seeking behavior, the need for developing more accurate document retrieval and ranking methods is ever more increasing. The recent trend in effective neural information retrieval is a testament to the importance of this topic [1]. In this context, the *ad hoc document retrieval* task has been one which has progressively received more attention and is focused on retrieving and ranking relevant documents to satisfy users' information needs, which are often expressed in the form of a short query. There have recently been a noticeable number of neural methods that have shown strong retrieval effectiveness for the ad hoc retrieval task [2,1,3–5]. However, despite their strong performance, researchers have reported that still, the increased performance of such methods is not uniform across all types of queries [6].

* Corresponding author.
E-mail addresses: m_khodabakhsh@shahroodut.ac.ir (M. Khodabakhsh), bagheri@ryerson.ca (E. Bagheri).

https://doi.org/10.1016/j.ins.2023.119015
Received 20 December 2022; Received in revised form 15 April 2023; Accepted 22 April 2023

Available online 28 April 2023

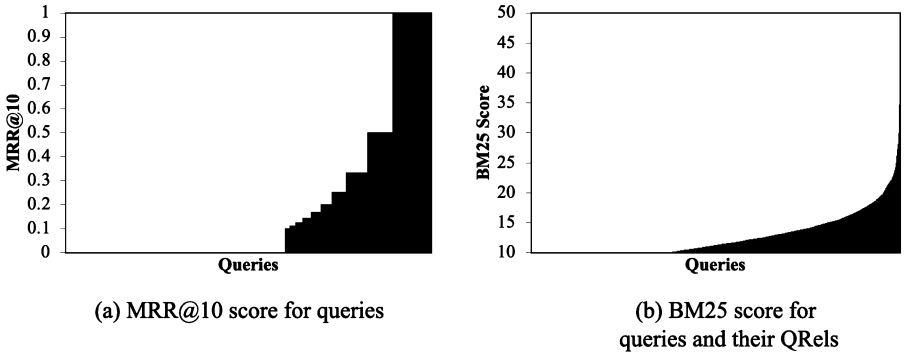

Fig. 1. The contrast between the MRR@10 score of queries on the left (a) and the BM25 score of the queries with their related documents in the QRels on the right (b).

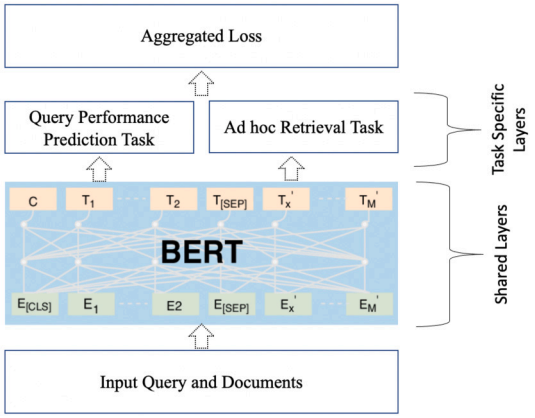

Fig. 2. Schematic View of our Proposed M-QPPF Framework.

For this reason, there have been works that aim at predicting how well a retrieval method is able to satisfy an input query This task is often referred to as *Query Performance Prediction (QPP)* [7–15]. The main objective of QPP methods is to estimate the quality of the ranked list of documents and its utility to satisfy user's information needs. A reliable QPP method can be effectively used for such purposes as reformulating the original query [16], or routing the query to alternative rankers, among others [17].

There is a clear interplay between the ad hoc retrieval and query performance prediction tasks where the performance of ad hoc retrieval methods impacts how well QPP methods are able to predict query performance. Despite the close connection between the two tasks, existing methods for each of the tasks have treated them independently whereby methods have specifically been designed for ad hoc retrieval and query performance prediction in isolation. Our work is among the first, if not the first, to consider addressing both tasks in tandem. In other words, our main objective is to learn to perform ad hoc retrieval while at the same time learning to predict the quality of the performance of a query through a multi-task learning framework [18]. Our hypothesis is that learning to rank and learning to predict query performance simultaneously will result in more effective ranking and more accurate performance prediction given the synergies between the two tasks. This hypothesis is primarily motivated by earlier work in the literature that has separately focused on query performance prediction as well as ad hoc retrieval tasks. In the context of query performance prediction, various researchers have argued and empirically shown that effective pre-retrieval QPP methods would need to consider features that are used in retrieval methods. For instance, the NQC method [10] uses a normalized variance of retrieved document scores produced by the retrieval method as a measure for query performance. Similarly, the SMV method [11] employs both the variance of the distribution of the relevance scores among the retrieved documents provided by the retrieval method as well as their magnitude as the indicator of query performance. On the other hand, there have also been works in the literature that have used QPP methods for improving retrieval performance through query routing [19] and quality prediction [16]. Therefore, as indicated by the related literature, there is a noticeable synergy between the two tasks.

Furthermore and beyond the literature, we highlight that such a synergy between the two tasks can also be observed empirically. To visualize this synergy, we plot the average BM25 score of the retrieved documents per query as well as the Reciprocal Rank (RR) score for each query of the MS MARCO Dev set in Fig. 1. These two sub-diagrams exhibit similarities between the distribution of BM25 scores and RR scores over all queries, which is an indication that the distance between the query and its relevant documents, and the effectiveness of the retrieved documents for the input query have very similar patterns.

For this reason, we propose the Multi-task Query Performance Prediction Framework, referred to as M-QPPF, which jointly learns to rank documents and predict the quality of the retrieved list for a given query. As illustrated in Fig. 2, we formulate the

multi-task learning process by fine-tuning a shared pre-trained BERT-based language model [20] based on ad hoc retrieval and QPP tasks in order to capture the semantic interactions between documents and queries. More specifically, we fine-tune BERT by adding task-specific layers on top of BERT where each task is learned through independent layers but shares similar fine-tuned BERT representations. Therefore, each task benefits from independent layer weights on top of BERT while sharing weights through the fine-tuned BERT.

The major distinguishing aspect of our proposed M-QPPF framework lies in its ability to jointly learn ranking and quality prediction at the same time; hence, providing feedback from QPP to ad hoc retrieval, and vice versa, when learning representations. This has a reinforcing impact on the performance of both tasks, as one will increase the performance of the other given their tight interdependence. This is in contrast to the state-of-the-art methods that fine-tune large language models, such as BERT, for either ranking or query performance prediction.

We evaluate our proposed framework against the state-of-the-art methods for both QPP and ad hoc retrieval tasks using widely used gold standard datasets including MS MARCO and TREC DL. We note that while our work is trained to learn a joint representation for QPP and ad hoc retrieval, it is able to show notable performance on each task separately and in comparison to the state-of-the-art methods in each task. More specifically, we find that our work shows significantly better QPP performance compared to all baselines and over all of the datasets. Furthermore and for ad hoc retrieval, our approach is able to show competitive or better performance compared to the baseline methods.

The contributions of our work in this paper can be enumerated as follows:

- We identify conceptual synergies between the ad hoc retrieval and query performance prediction tasks, and propose to address both tasks in tandem, which can lead to performance improvements in both tasks given their complementary behavior;
- We propose a novel framework that jointly learns to perform the ad hoc document retrieval and query performance prediction tasks through a multi-task learning setting, which fine-tunes the transformer-based BERT language model. This multitask learning framework appreciates the synergies between the two tasks through shared BERT layers and captures the specificities of each task through task-specific layers;
- We perform extensive experiments to show the impact of multi-task learning on predicting the performance of queries and ranking documents. Our experimental results show the effectiveness of our proposed framework compared to the state-of-the-art methods in each task.

The rest of the paper is organized as follows: in the next section, we review the related work in three subsections covering pertinent work in query performance prediction, document ranking, and multi-task learning applied to information retrieval. Sections 3 and 4 provide preliminaries, problem definition, and our proposed framework. The details of our experiments, datasets, evaluation metrics, baselines, and our findings are presented in Section 5. Finally, we conclude the paper in Section 6.

2. Related work

The work in this paper touches on two main tasks within information retrieval, namely query performance prediction and ad hoc retrieval, and also benefits from methods in multi-task learning. We will cover relevant literature in each of these three areas in this section.

2.1. Query performance prediction

The QPP task aims at estimating the quality of retrieval results without accessing relevance judgments. There are two main approaches for QPP, namely pre-retrieval and post-retrieval performance prediction. The approach taken by pre-retrieval predictors is to predict query performance based solely on the query and corpus statistics [21]. In contrast, post-retrieval predictors base their predictions on not only the query and the corpus but also on the ranked list of documents retrieved by the retrieval method. More traditional post-retrieval predictors often operate based on a function that exhibits an association between the query and document spaces based on features extracted from the query, corpus, and the initial list of ranked documents for the query. These features include the distribution of the retrieved document scores [9–11], the degree of difference between the characteristics of the retrieved documents and those of the entire corpus [8], and the robustness of the retrieved documents against perturbations in the query [9], the result list [22], and the retrieval method [23], among other methods.

Several researchers have recently studied the impact and importance of neural techniques in the QPP task [24,19]. Arabzadeh et al. [19] introduced a set of pre-retrieval QPP metrics based on pre-trained neural embeddings and showed that they are more effective than the known pre-retrieval QPP metrics. In another work [24], the authors proposed three neural features that employ neural embeddings to capture the semantic relationship between the representation of documents and queries based on their relationship in the embedding space and then integrated them into existing traditional post-retrieval predictors.

In addition to the above pre-retrieval predictors, there are other post-retrieval predictors based on neural architectures that estimate the performance of queries in a classification or regression paradigm. Zamani et al. [12] were among the first to use a weak supervision strategy, a supervised learning approach that obtains training labels from existing unsupervised performance predictors, to propose a QPP model with three neural components: (1) a component to represent the top-k retrieval scores, (2) another for representing the term distribution in the top retrieved documents, and (3) the last one to represent the documents in semantic space. Similarly, Arabzadeh et al. [25] proposed a method which learns to directly predict a retrieval score such as MRR or MAP instead of

predicting a query performance score. This method uses two widely-used neural architectures, namely cross-encoder and bi-encoder architectures. Unlike the work by Arabzadeh et al., which estimates the performance of a query by only considering the first retrieved document in the results list, Chen et al. [15] have introduced a BERT-based Group-wise predictor where the ranking contexts of a list of queries are jointly modeled to predict the relative performance of individual queries.

While methods such as Neural-QPP and BERT-based Group-wise predictor are based on regression models, a more recent BERT-based predictor, namely qppBERT-pl [14], transforms the QPP objective into a classification task and is trained based on a pointwise loss function on each individual query as well as a listwise loss over the top-ranked documents (split into chunks) to predict the number of relevant documents in each chunk for a given query.

One of the downsides of traditional post-retrieval QPP methods is that they rely on term statistics within both the query and document spaces; hence, they do not necessarily perform as effectively when a vocabulary mismatch exists between the two spaces. In order to address the vocabulary mismatch problem, Zamani et al. [12] have adopted the neural embedding technique to represent both queries and documents and proposed the NeuralQPP method. While addressing vocabulary mismatch, NeuralQPP may not be able to distinguish between potentially differing semantics associated with the same surface form of a term as they are context-independent. For this reason, Arabzadeh et al. [25] proposed BERT-QPP that employs contextualized pre-trained embeddings such as BERT [20] for the QPP task. BERT-QPP is however limited to only considering the first retrieved document in the results list. Unlike BERT-QPP, qppBERT-pl [14] uses the top-ranked documents and splits them into chunks that cause the loss of semantics and coherency in the documents. The advantage of our work over existing QPP methods is that our work considers the list of top-ranked documents in which each document along with the query is encoded through BERT.

2.2. Ad hoc document ranking

Most traditional ad hoc document ranking methods, often known as sparse ranking methods, such as BM25 and query likelihood, rely purely on lexical matching between the terms of the query and the documents. Given their reliance on term matching, these methods often suffer from issues such as vocabulary mismatch, referring to situations where there is little lexical overlap between the query and its relevant documents. More recent methods have capitalized on the representation power of large pre-trained neural language models [26,27]. The neural models are often developed by fine-tuning existing language models, such as BERT [20]. Neural retrieval methods can broadly be categorized into representation-based and interaction-based methods. In representation-based methods, queries and documents are first separately embedded into continuous vectors. The representations are then tuned to minimize the distance between the representation of a query and its relevant documents and maximize its distance from irrelevant documents [28–30]. On the other hand, interaction-based methods capture term interactions through cross-match attentions between the query and document terms [31,32].

In general, studies have shown that neural methods that adopt an interaction-based paradigm are more effective but slower than their counterparts in the representation-based paradigm [5]. Therefore, researchers have looked at ways through which interaction-based and representation-based methods could be integrated [33,2]. MacAvaney et al. [33] proposed the Precomputing Transformer Term Representations (PreTTR) method, which borrows concepts from both representation-based and interaction-based paradigms. PreTTR considerably reduces query-time latency of deep transformer networks making these networks more practical to use in a real-time ranking scenario. ColBERT [2] also benefits from both paradigms where it separately encodes the queries and documents using BERT and then evaluates their relevance in a cheap yet powerful interaction step.

The main disadvantage of some of the existing BERT-based rankers [28–30] is that the representation of the queries and documents in the embedding space is updated separately during the fine-tuning process. Therefore, the representations of the query and documents are not dependent on each other and the performance of the ranking would not be as optimal compared to when both embeddings are fine-tuned in tandem [5]. In order to address the above problem, interaction-based methods [33,2] consider the association between the terms of both queries and documents to represent them in the same embedding space that can lead to higher effectiveness. As a disadvantage, it can be said that such methods would need to use a pointwise learning paradigm. The advantage of our work is that the listwise learning to rank paradigm is used to learn the relevance scores of documents to queries.

There is an abundant number of impactful research work on neural ranking. We encourage the interested reader to explore further works in [34].

2.3. Multi-task learning

While multi-task learning has seen wider application in the broader machine learning community [35], there have also been a number of works in the context of ad hoc retrieval that have adopted the multi-task learning strategy to learn other tasks in tandem with the ranking task [36–39]. For instance, Liu et al. [36] proposed a multi-task deep neural framework to unify the representation learning process for query classification and document retrieval.

Based on the fact that document ranking and next query prediction are both core tasks in session-based search and are often influenced by the same search intent, recent works have attempted to address both tasks through multi-task learning. Methods such as M-NSRF [37], GDMTL [38], and LostNet [39] are multi-task learning frameworks that learn to rank documents and predict the next query in tandem. More specifically, M-NSRF adopts a sequence-to-sequence architecture and demonstrates that document ranking and query suggestion tasks benefit from each other by sharing session-level latent recurrent states [37]. Similarly, Cheng et al. [39] have proposed LostNet in order to additionally capture users' long-term intentions. To do so, LostNet adopts a hierarchical session-based attention mechanism for inferring the intention of each session, a multi-hop external memory network to learn users'

long-term intentions and a joint learning framework that optimizes for both document ranking and query prediction tasks. GDMTL is another framework that integrates generative and discriminative tasks by exploiting readily available generative tasks such as query generation to improve the performance of discriminative ranking models [38].

Our work distinguishes itself from the above multi-task learning approaches in that it learns to perform query performance prediction and ad hoc retrieval tasks in tandem, which has not been explored in the past. In addition since (1) interaction-based neural methods have generally shown to outperform representation-based methods for the ad hoc retrieval task [5] and (2) BERT-based post-retrieval methods [14,13] have shown strong performance for the QPP task, we benefit from both interaction-based retrieval methods and BERT-based post-retrieval QPP methods to construct a representation that can be shared across both tasks and to effectively capture deeper contextual associations between the query and document spaces.

3. Problem definition

The objective of our work is to address the tasks of ad hoc document retrieval and query performance prediction in tandem through a multi-task learning approach. We first formally define each of these tasks before presenting the details of our proposed approach.

Ad hoc document retrieval is one of the main tasks in information retrieval, which is focused on identifying and rank-ordering a list of relevant documents from a document corpus for a given information need often expressed in the form of a query. Let C and q denote a document corpus and an input search query, respectively. The *ad hoc document retrieval task* can be formally defined as follows:

$$D_q \leftarrow R(q, C) \tag{1}$$

where $R()$ is a function that takes C and q as input and returns a ranked list of documents, D_q. As a result of the ranking, every document in the corpus will receive a score that can be used to measure the relevance of a document to the query. In order to evaluate the performance of $R()$, gold standard datasets, known as relevance judgment sets, are used which provide a set of queries with their relevant documents with varying degrees of relevance. An effective function $R()$ would be one that is able to identify all relevant documents to q (perfect recall) while ranking all the retrieved relevant documents at the top of the ranked list of documents (perfect precision). The quality of the results D_q produced by $R()$ is often measured by ranking metrics such as Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG), among others.

The success of the ad hoc retrieval task and hence users' satisfaction, relies on the quality of the results D_q produced by $R()$ [7]. Hence, it is essential to estimate the quality of D_q in order to proactively determine whether the results will satisfy the user's information needs or not. As such, given the user query q, and a list of ranked documents D_q produced by a ranking method $R()$, the objective of the query performance prediction task is to predict one of the quality indicators for D_q, e.g., MAP. The *query performance prediction* task can be formally defined as follows:

Given D_q returned by $R()$, the goal of the QPP task is to offer a predictor $\mu()$, for estimating the performance of $R()$ for q based on an evaluation metric M:

$$\widehat{M} \leftarrow \mu(q, D_q) \tag{2}$$

The association between M and $\widehat{M}$ is often considered as the usefulness of $\mu()$. We will review approaches for evaluating $\mu()$ in the experiments section of this paper. With the above formal definitions for ad hoc retrieval $R()$ and query performance prediction $\mu()$, the objective of our work in this paper is to develop a multi-task learning framework, namely M-QPPF, that would jointly learn to maximize the performance of $R()$ and $\mu()$.

4. Proposed approach

Our proposed M-QPPF approach is based on multi-task learning for the ad hoc retrieval and QPP tasks. We first provide the overall model of how these two separate tasks are integrated within multi-task learning and then present the details of each component of M-QPPF separately.

4.1. The proposed M-QPPF architecture

Our proposed M-QPPF approach combines ad hoc retrieval (document ranking) and QPP tasks using multi-task learning. The overall architecture of M-QPPF is shown in Fig. 3. In our proposed architecture, we would like the model to be able to learn the similarities between the two tasks through shared components but at the same time also learn the specificities of each of the tasks in isolation. To accommodate this, our architecture consists of a shared BERT layer, which will be fine-tuned to capture the similarities between the two tasks. Furthermore and in order to capture the specifics of each task, we mount two separate layers on top of BERT to learn the details of the QPP and document ranking tasks. The parameters of the shared BERT layer are trained by the multi-task objective to capture the shared characteristics of the two tasks, while the parameters of the mounted layers are learned by optimizing the task-specific objectives [35].

We would like to highlight how the shared characteristics of the two tasks are captured by our proposed approach. As seen in Fig. 3, each of the two tasks has task-specific layers that capture the detailed characteristics of each task. However, given the multi-task context, our model will need to also capture commonalities. We capture such commonalities between the two tasks by using

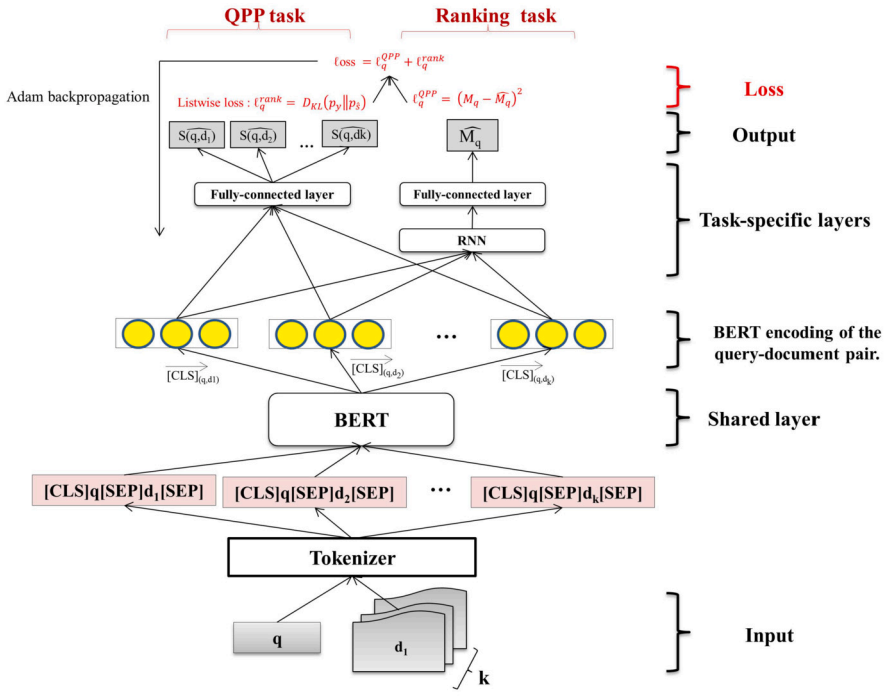

Fig. 3. The Architecture of our Proposed M-QPPF Approach.

a shared BERT layer. This shared layer is fine-tuned for these two tasks simultaneously through an aggregated loss function that backpropagates the loss from each of the two tasks. As such, the shared fine-tuned BERT will capture the common characteristics of the two tasks.

More specifically, M-QPPF simultaneously optimizes two different loss functions, one loss function for the document ranking task and another for the QPP task. Let (q, D_q, y, M_q) be a training sample where D_q denotes the set of ranked documents retrieved for query q, y be a set of relevance judgments, such that the i^{th} element represents the relevance judgment for the query-document pair (q, d_i), and M_q be the value of an evaluation metric. Using multi-task learning, we define the loss function for every training sample as follows:

$$\ell^q = \ell^q_{Rank} + \ell^q_{QPP} \tag{3}$$

where ℓ^q_{Rank} is the loss function to train for the document ranking task and ℓ^q_{QPP} is the loss for the QPP task. The loss over a set of queries Q is defined as the average losses:

$$L = \frac{1}{|Q|} \sum_{q \in Q} \ell^q \tag{4}$$

4.2. Ad hoc retrieval task

As already extensively explored in the literature [2,28,29], pre-trained transformer-based language models, such as BERT [20], show strong performance on the ad hoc retrieval task. Given language models such as BERT are trained for tasks such as Next Sentence Prediction (NSP), they include additional meta-tokens that often capture the semantics of a sequence of terms including the [SEP] and [CLS] meta-tokens [20]. Earlier work has shown that the meta-tokens could be fine-tuned for performing downstream tasks such as document ranking [28,29], and text classification, among others. For instance, MacAvaney et al. [29] and Nogueira et al. [28] introduced *BERT-based ranking* where the query-document pairs are considered as two text sentences, which are separated by the [SEP] meta-token and whose relevance relation can be captured through a leading [CLS] meta-token. We adopt a similar strategy to model the ranking task mounted on top of BERT.

For a given input query q and a document d, we first feed the concatenation of the query and document pair through BERT, and then the contextual representation of the leading [CLS] meta-token is used as input to a fully connected layer to predict the relevance scores for the query-document pair as follows:

$$\overrightarrow{[CLS]}_{(q,d)} = BERT([CLS] \quad q \quad [SEP] \quad d \quad [SEP]) \tag{5}$$

$$\widehat{s(q,d)} = \phi(W.\overrightarrow{[CLS]}_{(q,d)} + b) \tag{6}$$

where $\overrightarrow{[CLS]}_{q,d}$ is the final hidden states of the [CLS] meta-token and denotes the BERT encoding of the query-document pair (q,d). The fully-connected layer, also known as the hidden layer, is a combination of the affine function and the activation function. The fully-connected layer takes input from the previous layer which is a one-dimensional input. The input is passed first to the affine function (multiplies the input by a weight matrix and then adds a bias vector) and then to the activation function (such as linear, sigmoid, etc). W is the weight matrix, b is the bias term, $\phi(.)$ is the linear activation function for the fully connected layer, and $\widehat{s(q,d)}$ is the predicted relevance score.

Learning the parameters of the ranking model can be accomplished using a listwise learning to rank paradigm such as ListNet [40] whose main goal is to represent the difference between the predicted score list by the ranking model and the relevance label list given as ground truth. To this end, both lists of predicted relevance scores and relevance labels are transformed into probability distributions through a softmax function, and then the difference between the two distributions is measured through Kullback-Leibler divergence [41] as the loss function:

$$\mathscr{L}^{q}_{Rank} = D_{KL}(p_y||p_{\hat{s}}) = \sum_{i=1}^{k} p_y(q,d_i) log \frac{p_y(q,d_i)}{p_{\hat{s}}(q,d_i)} \tag{7}$$

in which p_y and $p_{\hat{s}}$ are the probability distribution of the relevance label list and predicted score list respectively, and could be understood as encoding the likelihood of document d_i appearing at the top of the ranked list, referred to as *'top one'* probability, according to the labels and scores, respectively [40]:

$$p_y(q,d_i) = \frac{y_(q,d_i)}{\sum_{j=1}^{k} y_(q,d_j)} \quad , \quad p_{\hat{s}}(q,d_i) = \frac{\widehat{s(q,d_i)}}{\sum_{j=1}^{k} \widehat{s(q,d_j)}} \tag{8}$$

Here, $y_(q,d_i)$ and $\widehat{s(q,d_i)}$ are the truth label and the predicted score of d_i for given query q, respectively; and, k is the number of retrieved documents in the list. Our main reason for the choice of the listwise loss function is that QPP methods need a list of documents that are sorted by a ranking model to enable the prediction of the query performance. Hence, given we are interested in designing a multi-task learning approach our choice for the listwise loss function in the ad hoc retrieval task is driven by the requirements of the QPP task.

4.3. Query performance prediction task

Similar to the ad hoc retrieval task, recent studies on the query performance prediction task have shown that BERT could be fine-tuned in different ways to serve as a strong indicator for query difficulty [14,15]. BERT-based QPP methods often encode query-document pairs in the context of BERT in a similar fashion to the encoding performed by methods in the ad hoc retrieval task as discussed in Equation (5). As such, given the similar encoding between the tasks for query-document pairs, one could easily identify this as a way to integrate the two tasks within a multi-task learning framework where the encoding of the query-document pairs, expressed as $\overrightarrow{[CLS]}_{(q,d)}$, would serve as a shared knowledge that can be transferred between the two tasks.

In order to establish the performance prediction function $\mu()$, as defined in Equation (2), for a given query q and a list of top-k ranked documents D_q, we feed each (q,d_i) pair into BERT according to Equation (5) in order to produce the encoding $\overrightarrow{[CLS]}_{(q,d_i)}$ as its representation. This encoding is then fed into a recurrent neural network as follows:

$$h_i = RNN(h_{i-1}, \overrightarrow{[CLS]}_{(q,d_i)}) \quad ; \quad i \in \{1,2,...,k\} \tag{9}$$

where RNN can be any recurrent neural network module such as LSTM [42] or GRU [43]. In this equation, h_i is the hidden state of the RNN, which represents D_q at rank i. In other words, at rank i, the RNN reads the encoding of (q,d_i) and updates its hidden state h_i.

There are two main reasons for adopting the recurrent neural network architecture for the query performance prediction task. Our first reason is that D_q consists of a ranked list of documents, which means that the order of the documents is important. Despite some post-retrieval methods such as the *clarity* method that compute the QPP score based on the occurrence frequency of the corpus terms in D_q and do not consider the order of documents in D_q [8], we propose that capturing the order of documents in the final representation of D_q will allow us to capture additional information about query difficulty. As an example, consider two lists of length five both of which consist of the relevant document but one places the relevant document at rank one and the other at rank five. Clearly, these two ranked lists of documents and their associated queries do not have the same degree of difficulty, as such, a recurrent neural network architecture will allow us to capture such differences. The second reason is that a recurrent architecture allows us to systematically aggregate information from different documents in D_q instead of using fixed aggregation methods such as max-pooling or average-pooling [42,43]. In other words, the recurrent architecture will produce a single h_k representation for the top-k documents retrieved and ranked for the given query q through a systematic recurrent approach.

Given h_k as the representation generated by RNN for D_q, two fully-connected layers with a non-linear activation function are used to predict the QPP score for query q:

$$\widehat{M(q,D_q)} = \sigma(W_2.\phi(W_1.h_k + b_1) + b_2) \tag{10}$$

where $\sigma()$ is the sigmoid activation function, W_1, and W_2 are weighted matrices for the first and second fully-connected layers respectively, b_1, and b_2 are biases and $\widehat{M(q,D_q)}$ is the predicted QPP score.

Table 1
Statistics of the datasets used in our experiments.

	#Queries	#QRel
MS MARCO Train set	502,939	532,761
MS MARCO Dev set	6,980	7,437
TREC DL 2019	43	9,260
TREC DL 2020	54	11,386
DL-Hard	50	4,256
Corpus	8,841,823 passages	

Since the QPP task can be viewed as a regression problem [25] and given a tuple of training data (q, D_q, M_q), where M_q is the value of an evaluation metric such as MAP, the objective of the QPP task would be to minimize the squared error between the predicted QPP score, $\widehat{M(q, D_q)}$, and M_q. More formally, the loss function for the QPP task can be written as:

$$\ell^q_{QPP} = (M_q - \widehat{M(q, D_q)})^2 \tag{11}$$

In summary and as shown in Fig. 3, our proposed multi-tasking strategy is able to: (1) capture common characteristics of the two ad hoc retrieval and query performance prediction tasks through the shared fine-tuned BERT model. This is made possible by the fact that both tasks are focused on learning associations between query document pairs. (2) capture task-specific characteristics through the layered components on top of the shared BERT layer, which gives it access to common knowledge between the two tasks from the shared BERT with the possibility of learning additional task-specific parameters.

5. Experiments

In this section, we first introduce the datasets used in our experiments and then describe the evaluation metrics, baselines, and details of our experimental setup. Finally, we report and analyze our findings in detail.

5.1. Datasets

The MS MARCO[1] passage collection [44] is a common and well-known dataset in both ad hoc retrieval and QPP tasks. MS MARCO (Microsoft Machine Reading Comprehension) includes 8.8M passages extracted from web documents and over 500k search queries sampled from the Bing search engine logs. The MS MARCO training set contains around 532k relevant judgments, which means that there is on average one relevant passage marked by human assessors per query. We used these queries along with their relevance judgments to train M-QPPF while we evaluated it on a completely non-overlapping query set called the MS MARCO Development set (Dev set, for short), which consists of 6, 980 queries and their relevant passage pairs. In addition to the Dev set and for more in-depth evaluation, we also additionally use the TREC Deep Learning Track 2019 [45], 2020 [46], and Deep Learning Hard (DL-Hard) [47] query sets with 43, 54, and 50 queries, respectively in our evaluations. In contrast to the MS MARCO Dev set, which has around one relevant document per query, these three query sets provide multiple relevant documents per document judged on a 4-level relevance scale. Summary statistics for the datasets used in our experiments are shown in Table 1.

5.2. First stage retriever

Similar to prior work [14], we selected BM25 as the first stage retriever to identify a list of the top-1000 passages per query. In order to perform experiments on the *ad hoc retrieval task*, we re-rank the top-1000 passages retrieved by BM25 using the ad hoc retrieval component of M-QPPF as well as the other state-of-the-art ad hoc retrieval baselines. Similarly, when evaluating our work against the state-of-the-art on the *QPP task*, we use the top-1000 passages retrieved by BM25.

5.3. Evaluation metrics

Ad hoc retrieval metrics. For the purpose of evaluating our proposed M-QPPF approach for ad hoc retrieval, we employed the official metrics for each dataset. The official metric for MS MARCO is MRR@10 [44]:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{12}$$

where $rank_i$ refers to the rank position of the first relevant document for the i-th query.

The official metric for TREC DL 2019 and 2020, as well as DL-HARD is nDCG@10 [45,46]:

$$NDCG = \frac{DCG}{IDCG} \tag{13}$$

[1] https://microsoft.github.io/msmarco.

$$DCG = \sum_{d \in D_q} \frac{2^{rel(d)} - 1}{log_2(rank_d + 1)} \tag{14}$$

where $rel(d)$ and $rank_d$ are the graded relevance of document d and its rank in the result list, respectively. $IDCG$ is computed the same way but by assuming an ideal rank order for the documents.

QPP metrics. We measure the correlation between the set of queries that are ranked based on their predicted performance against their actual performance (MRR@10 or NDCG@10, depending on the dataset) in order to evaluate MQPPF for the QPP task. To this end, we used Pearson ρ, Kendall's τ, and Spearman ρ coefficients to report the ability of M-QPPF in predicting query performance. Pearson ρ coefficient as a linear correlation metric is the ratio between the covariance of two lists and the product of their standard deviations:

$$\rho_{M,\widehat{M}} = \frac{cov(M, \widehat{M})}{\sigma_M \sigma_{\widehat{M}}} \tag{15}$$

where $cov(M, \widehat{M})$ is the covariance, σ_M is the standard deviation of the actual performance of a list of queries, and $\sigma_{\widehat{M}}$ is the standard deviation of the predicted performance of the list of queries.

Kendall's τ coefficient, as a ranking correlation metric, measures the similarity of the orderings of the queries when ranked by their actual and predicted performance:

$$\tau = \frac{\text{number of concordant pairs - number of discordant pairs}}{\text{number of pairs}} \tag{16}$$

Spearman ρ is a ranking correlation metric and is defined as the Pearson correlation coefficient between the ranked queries:

$$\rho_{M,\widehat{M}} = \rho_{R(M),R(\widehat{M})} = \frac{cov((R(M), R(\widehat{M}))}{\sigma_{R(M)} \sigma_{R(\widehat{M})}} \tag{17}$$

where $R(M)$ and $R(\widehat{M})$ are the list of queries that are ranked based on their actual and predicted performance, respectively. The higher the correlation value is, the more accurate the predicted performance is.

5.4. Baselines

In this work, we selected two groups of baselines related to the two tasks.

5.4.1. Ad hoc retrieval baselines

We compare our work with the state-of-the-art methods which utilize BERT for ad hoc retrieval:

BERT base [28] is one of the first BERT-based methods where query and document input sequences are concatenated with special tokens and then the [CLS] vector computed by BERT is fed to a single-layer neural network to obtain the probability of the document being relevant. In our experiments, we set the values of hyperparameters $\beta 1$, $\beta 2$, $L2$ weight decay, and initial learning rate to 0.9, 0.999, 0.01, and $3 * 10^{-6}$, respectively.

TwinBERT [31] produces the [CLS] vector for query and document independently via two BERT-like encoders and then combines the two vectors through a crossing layer to compute the final similarity scores. The values of TwinBERT hyperparameters are similar to BERT base [28] except the learning rate that was set to $1e-4$.

monoT5[2] [3] is a pointwise re-ranker that uses T5, a popular pretrained sequence-to-sequence transformer model. In monoT5, the model is fine-tuned to produce the token 'true' or 'false' depending on whether the document is relevant or not to the query. At inference time, to compute probabilities for each query–document pair softmax is applied only on the logits of the 'true' and 'false' tokens.

ColBERT [2] encodes each query and document into a bag of contextual vectors using a shared BERT-based encoder which distinguishes the input sequence of the queries and documents by adding special meta-tokens [Q] and [D] to them, respectively. The relevance score of the document to the query is estimated via late interaction between their bags of contextualized vectors as a summation of maximum similarity (MaxSim) operators. The values of 128 and 32 were considered for the hyperparameters of ColBERT embedding dimensions and the number of embeddings per query at N_q, respectively.

PreTTR [33] improves the performance of BERT-based ranking models at query time by preprocessing the documents in the corpus in such a way that document term representations are computed partially through BERT up to the transformer layer and stored in an index. In the training stage, BERT is fine-tuned for ranking by masking attention scores in the first l layers, disallowing interactions between the query and the document. At query time, the query is processed through the first l layers and then combined with the document term representations to calculate the ranking score. In our experiments, the batch size was 16 pairs of relevant and non-relevant documents with gradient accumulation.

CoCondenser[3] [48] is an unsupervised corpus-aware language model pre-training method, which acquires two important properties of noise resistance and structured embedding space for dense retrieval by leveraging the Condenser architecture and a

[2] https://github.com/castorini/pygaggle.
[3] https://github.com/luyug/Condenser.

corpus-aware contrastive loss. The parameters of CoCondenser were estimated using stochastic gradient descent with AdamW for optimization.

SPLADE-CoCondenser-EnsembleDistil[4] [1] is an expansion of the SPLADE [49] ranker where SPLADE has been initialized from a pre-trained CoCondenser [48] checkpoint and then a distillation strategy is leveraged for training to improve the effectiveness of the ranking task. SPLADE-CoCondenser-EnsembleDistil has been initialized from a pre-trained CoCondenser checkpoint.[5]

5.4.2. QPP baselines

Given our work in M-QPPF falls within the realm of post-retrieval QPP methods, we compare our work against both unsupervised and supervised post-retrieval QPP methods:

Clarity [8] is one of the early methods in QPP, which measures the coherence of term distribution in the retrieved list of documents with respect to the corpus. Clarity computes the difference between the language model of the retrieved list of documents and the corpus through Kullback-Leibler divergence. The λ hyperparameter was set to 0.9.

WIG [9] is a score-based method that computes the mean divergence of the retrieval score of the top-ranked documents from that of the corpus in order to predict query performance.

NQC [10] is also a popular score-based method that calculates the standard deviation of the rank scores of the retrieved documents normalized by the corpus score.

SMV [11] is another score-based method, which estimates query performance by using both the magnitude and variance of the scores of the retrieved documents.

NeuralQPP [12] is the first QPP method to use existing unsupervised QPP methods as signals to perform weakly- supervised learning. According to [12], we initialized the embedding matrix by pre-trained vectors trained on Wikipedia dump 2014.

NQA-QPP [13] has been motivated by NeuralQPP but uses a BERT-base model that predicts query performance in non-factoid question-answering systems. Unlike NeuralQPP, NQA-QPP utilizes BERT in order to learn the representations from the question content and the top-ranked answers in the two last components.

qppBERT-PL[6] [14] is a BERT-based method that uses pointwise training on individual queries, but listwise over the top-ranked documents (split into chunks). The qppBERT-PL method considers the top-retrieved documents as a sequence, thus takes into account the relative positions (or ranks) of the top-k documents. Unlike NeuralQPP and NQA-QPP which use a regression model to learn the performance of the query, qppBERT-PL transforms the objective into a classification task, where the number of relevant documents in the top retrieved documents is predicted. The value of chunk per query was set to 5 in the experiments.

5.5. Implementation details of M-QPPF

Our proposed model was trained in an end-to-end fashion whose parameters were estimated using stochastic gradient descent with Adam for optimization. A 1-layer RNN with 768 hidden units was employed which means that D_q is represented as a vector with 768 dimensions. The dimension of fully connected layers was 100 (thus W, W_1, and W_2 should have a shape of 768 $\times$ 100). The learning rate was fixed at $2e-5$.

In our experiments, we used the pre-trained BERT [20] from HuggingFace with twelve layers, 768 hidden layers, twelve heads, and 110M parameters trained on lower-cased English text, often known as BERT-Base Uncased. The value of k, the number of documents in the retrieval list, was considered from {5, 10, 20} for both our work and the baselines. Based on this configuration, we used the top-5, 10, or 20 documents retrieved by BM25 in the MS MARCO dataset for the QPP and ad hoc retrieval tasks. As suggested by [14], we trained M-QPPF and BERT-based QPP baselines for one epoch on the queries of the MS MARCO training set.

Taking into account the values of the hyperparameters, M-QPPF with LSTM has 114M (114,284,746) parameters whose values are determined at training time. Nearly 110M (109,483,009) out of 114M parameters belong to the shared layers of BERT. Given that we have an LSTM module with one recurrent layer and two fully connected layers in the QPP task, the total trainable parameters in QPP-specific layers are 4M (4,801,737). The rest of the parameters belong to the ranking-specific layers, a fully connected layer With less than 1M (769k) parameters. The summary of the experiential environment configuration is listed in Table 2.

We record the validation loss of M-QPPF with the LSTM module during training when k is set to 5 and plot them in Fig. 4. The horizontal axis shows the number of samples that are used for training. As shown in the figure, we can observe that the validation loss converges at end of the training. Also, the training lasts nearly seven hours (401 minutes) for one epoch on NVIDIA RTX A6000. The training time is increased to 679 and 1,241 minutes by changing the value of k to 10 and 20, respectively. The details of the training time of M-QPPF are reported in Table 3.

Source Code. The data and code for this paper are publicly available for the sake of reproducibility.[7]

[4] https://github.com/naver/splade.
[5] https://huggingface.co/Luyu/co-condenser-marco.
[6] https://github.com/suchanadatta/qppBERT-PL.git.
[7] https://github.com/m-khodabakhsh/M_QPPF.

Table 2
The summary of the experiential environment configuration.

The number of epoch	1
Batch	16
Optimizer	Adam
Learning rate	$2e-5$
RNN (LSTM or GRU)	1-layer with 768 hidden units
The shapes of Weighted matrices	768×100
The shapes of biases	100
Pre-trained language model	BERT-Base Uncased
k	5, 10, 20
Total number of parameters	114M
GPU	NVIDIA RTX A6000

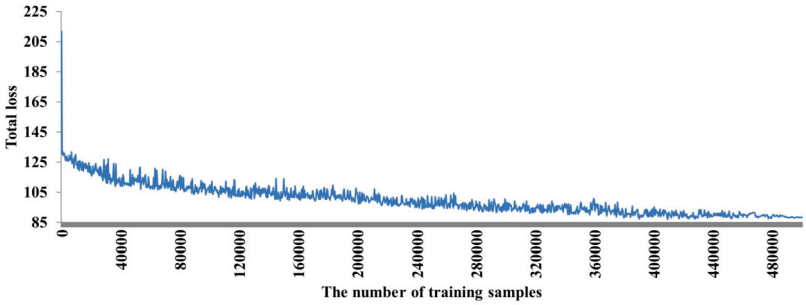

Fig. 4. The loss of validation set in training time.

Table 3
The training time of M-QPPF in minutes.

	k = 5	k = 10	k = 20
M-QPPF (LSTM)	401	679	1,241
M-QPPF (GRU)	426	700	1,267

5.6. Evaluation results

In this section, we first explore the influence that the choice of the recurrent neural network unit has on the overall performance of M-QPPF. Subsequently, we compare our proposed approach with the state-of-the-art baselines on the ad hoc retrieval and QPP tasks.

5.6.1. Impact of the RNN unit on the performance of M-QPPF

As outlined in Fig. 3, we capture the impact of the order of documents through a recurrent neural network; therefore, the choice of the RNN unit may possibly influence the overall performance. For this reason, we investigate the impact of this choice by comparing the performance of M-QPPF for cases when LSTM [42] and GRU [43] units are used for implementing the RNN component. Prior work has shown that a GRU unit can achieve competitive performance to an LSTM while requiring fewer parameters; hence, allowing it to train faster and generalize with less data [50]. We report the comparative results of the impact of the LSTM and GRU units on the overall performance of M-QPPF in Tables 4 and 5 for QPP and ranking tasks, respectively.

In Table 4, the performance of M-QPPF in terms of Pearson ρ, Kendall's τ, and Spearman ρ for different values of k are reported separately. The largest performance for each value of k and dataset is shown in bold. As shown in Table 4, there is no notable difference in the performance of M-QPPF when LSTM is used compared to when GRU is employed. The performance of the two units is highly competitive on all cases and the results of both units are statistically significant on the QPP task based on a two-tailed paired t-test with a confidence interval of 95% dataset for all values of k and all correlation metrics. Also, M-QPPF with GRU has the best performance on three out of four datasets when k is set to 5 and 20. In contrast, with k equal to 10, the M-QPPF with LSTM is more successful on three datasets of TREC DL 2019, TREC DL 2020, and DL-Hard.

To further investigate the impact of the choice of the recurrent unit, we explore the performance of M-QPPF on the ranking task. The results reported in this task are based on MRR@10 for the MS MARCO dataset and nDCG@10 for others, which are the official metrics for each of these datasets. As it is not possible to calculate the official metrics for k less than 10, we report the results of the experiments in which the number of top-k retrieved documents is 10 and 20. As seen in Table 5, the difference between the performance of M-QPPF with LSTM and GRU is minimal in most datasets, especially for k equal to 10. Overall, we conclude that the difference between LSTM and GRU is almost negligible on the ranking task except for the MS MARCO dataset where the performance of M-QPPF with GRU is substantially better than LSTM, by about 2.5% in terms of MRR@10.

Table 4
The performance of M-QPPF for the QPP task. * denotes statistically significant on MRR@10 with a two-tailed paired t-test at a 95% confidence interval.

		MS MARCO Dev set			TREC DL 2019			TREC DL 2020			DL-Hard		
		Pearson ρ	Kendall τ	Spearman ρ	Pearson ρ	Kendall τ	Spearman ρ	Pearson ρ	Kendall τ	Spearman ρ	Pearson ρ	Kendall τ	Spearman ρ
k = 5	M-QPPF (LSTM)	0.569*	0.419*	0.54*	0.274	0.298*	0.378*	**0.389***	**0.415***	**0.513***	0.374*	0.41*	0.519*
	M-QPPF (GRU)	**0.57***	**0.435***	**560***	**0.296**	**0.318***	**0.408***	0.387*	0.414*	0.499*	**0.452***	**0.443***	**0.565***
k = 10	M-QPPF (LSTM)	**0.596***	**0.446***	**0.573***	**0.314***	**0.408***	**0.504***	**0.391***	**0.396***	**0.486***	0.327*	0.35*	0.44*
	M-QPPF (GRU)	0.592*	0.443*	0.569*	0.296	**0.408***	0.494*	0.381*	0.371*	0.454*	**0.368***	**0.408***	**0.514***
k = 20	M-QPPF (LSTM)	**0.61***	**0.472***	**0.601***	0.335*	0.382*	0.473*	0.383*	0.389*	0.466*	0.249	0.318*	0.406*
	M-QPPF (GRU)	0.604*	0.47*	0.598*	**0.368***	**0.398***	**0.486***	**0.402***	**0.419***	**0.513***	**0.394***	**0.413***	**0.502***

Table 5
The M-QPPF Performance for the Ranking Task.

		MS MARCO Dev set	TREC DL 2019	TREC DL 2020	DL-Hard
		MRR@10	NDCG@10		
k = 10	M-QPPF (LSTM)	**0.276**	0.87	0.863	0.573
	M-QPPF (GRU)	0.273	**0.875**	**0.866**	**0.577**
k = 20	M-QPPF (LSTM)	**0.304**	**0.83**	**0.828**	0.509
	M-QPPF (GRU)	0.301	0.822	0.825	**0.518**

Based on the results shown in 4 and 5, we find that the choice of the recurrent neural network unit does not necessarily impact the overall performance of M-QPPF and slight performance variations can be observed between the two depending on the task and the dataset.

5.6.2. Effectiveness of M-QPPF for the QPP task

We compare the performance of our proposed M-QPPF method with the state-of-the-art baselines for the QPP task to understand whether our multitask learning framework leads to better overall performance on the query performance prediction task. The comparative results for the QPP task are reported in 6. Given the competitive nature of the GRU and LSTM units, we report the performance of M-QPPF based on both unit types in this table.

Based on the results, we observe that M-QPPF shows stronger results in predicting query performance on all four datasets for all values of k in terms of Pearson ρ, Kendall's τ, and Spearman ρ (except TREC DL 2019 where qppBERT-PL shows better performance in terms of Pearson ρ for k equal to 10 and 20). More concretely, and based on the MS MARCO Dev dataset, which is the dataset with the largest number of queries, and compared to the best performing QPP baseline, namely NQA-QPP, M-QPPF (w/GRU) shows an improvement of 18%, 16%, and 18% compared with NQA-QPP on various values of k in terms of Pearson ρ. Similar observations can be made based on M-QPPF (w/LSTM) on the different datasets and compared to the strong baselines.

An important observation that can be made based on the results in Table 6 is with regards to BERT-based QPP baselines. We observe that BERT-based baselines are often among the best-performing baselines regardless of the dataset or the evaluation metric. While contextualizing this finding with our proposed approach, one can conclude that the consideration of query semantics, as captured through BERT, plays a key role when predicting query performance. Those models, even when based on neural architectures such as NeuralQPP, that do not consider query semantics show notably weaker performance compared to BERT-based models.

Another observation is about the performance consistency of the qppBERT-PL baseline compared to the other baselines on three of the datasets, namely MS MARCO Dev, TREC DL 2019, and TREC DL 2020 when k is equal to 5. The reason for this may be that qppBERT-PL addresses the problem of QPP as a classification task and regards the total number of relevant documents that are observed in the set of top-k retrieved documents (here k = 5). For queries, the number of relevant documents in the top-k is not many and in several queries, the number of relevant documents in the top-k can be zero. Therefore, this will prevent classification-based QPP methods to work effectively on such queries; however, regression-based QPP methods are able to circumvent this issue even when there are zero relevant documents in the top-k. As such, one may conclude that QPP methods that are based on regression can be more effective than those that utilize a classification model to predict query performance for smaller values of k.

We would like to highlight that in contrast to the state-of-the-art baseline approaches, both variations of our proposed framework demonstrate stable performance on all four datasets and for all values of k, which means that M-QPPF does not show varying degrees of performance depending on the dataset or the depth of k. This is in contrast to the other baselines. For example, NQA-QPP is the best baseline on the MS MARCO Dev dataset for all values of k on all of the three correlation metrics while it does not show a strong performance on the TREC DL 2020 dataset at k equal to 5 or 10. Another example can be observed for qppBERT-PL as the worst and best baseline in TREC DL 2019 for k values of 5 and 10 respectively in terms of all correlation metrics. Also, as seen in the DL-Hard dataset, there is noticeable unstable behavior by the baselines for predicting query performance. For instance, SMV (as an unsupervised method) and NQA-QPP (as a supervised BERT-based method) are the strongest baselines for k equal to 5 and 10 respectively across all correlation metrics while the situation is completely different in the case of k equal to 20 where qppBERT-PL and NQC provide the best performance in terms of Pearson ρ, Kendall's τ, and Spearman ρ, respectively.

Overall, we find that our proposed M-QPPF method shows the best and most robust performance on the QPP task on a range of datasets and correlation metrics.

5.6.3. Effectiveness of M-QPPF for the ranking task

We also study the effectiveness of our proposed M-QPPF approach on the ranking task by comparing its final ranking performance with state-of-the-art baselines. Table 7 shows the retrieval performance of BM25 in addition to the re-ranking performance of M-QPPF and the baselines for various values of k across all evaluation datasets. Based on the results reported in the table, we observe that both variants of our proposed framework are able to improve the re-ranking performance significantly in the last three datasets, i.e. TREC DL 2019, TREC DL 2020, and DL-Hard in terms of NDCG@10. Especially in TREC DL 2019 and TREC DL 2020 where M-QPPF with GRU provides the greatest improvements over the best baselines equivalent to 32% and 34%, respectively when k is equal to 10.

Based on the reported performance of the baselines in the last three datasets, another important point that one could conclude is that the performance of our method is stable on the ranking task for all values of k. In TREC DL 2020, monoT5 is the best baseline for

Table 6
Comparison between the effectiveness of M-QPPF on the QPP task and the baselines. * denotes statistically significance with MRR@10 at 95% confidence interval using a two-tailed paired t-test.

		MS MARCO Dev set			TREC DL 2019			TREC DL 2020			DL-Hard		
		Pearson ρ	Kendall τ	Spearman ρ	Pearson ρ	Kendall τ	Spearman ρ	Pearson ρ	Kendall τ	Spearman ρ	Pearson ρ	Kendall τ	Spearman ρ
k = 5	Clarity	0.003	0.012	0.016	-0.094	-0.164	-0.213	0.056	-0.041	-0.05	0.144	0.001	-0.002
	WIG	0.118*	0.137*	0.182*	0.16	0.132	0.165	0.176	0.136	0.181	0.218	0.257	0.344
	NQC	0.045*	0.279*	0.369*	0.172	0.142	0.18	0.176	0.202	0.256	0.338	0.329	0.428
	SMV	0.043*	0.277*	0.368*	0.171	0.138	0.175	0.183	0.214*	0.263	0.349	0.333	0.433
	NeuralQPP	0.297*	0.266*	0.35*	0.233	0.228	0.297	0.352*	0.322*	0.421*	0.348	0.247	0.306
	NQA-QPP	0.387*	0.342*	0.424*	0.068	0.126	0.143	0.25	0.222*	0.268*	0.218	0.325	0.395
	qppBERT-PL	-0.019	-0.017	-0.019	-0.169	-0.071	-0.083	0.014	-0.022	-0.024	0.313	0.267	0.295
	M-QPPF (LSTM)	**0.569***	**0.419***	**0.54***	**0.274**	**0.298***	**0.378***	**0.389***	**0.415***	**0.513***	**0.374***	**0.41***	**0.519***
	M-QPPF (GRU)	**0.57***	**0.435***	**560***	**0.296**	**0.318***	**0.408***	**0.387***	**0.414***	**0.499***	**0.452***	**0.443***	**0.565***
k = 10	Clarity	0.003	0.012	0.016	-0.096	-0.146	-0.195	-0.003	-0.026	-0.028	-0.045	-0.048	-0.062
	WIG	0.083*	0.106*	0.141*	0.145	0.125	0.159	0.146	0.119	0.158	0.211	0.247	0.332
	NQC	0.034*	0.328*	0.406*	0.111	0.118	0.151	0.189	0.212	0.261	0.194	0.277	0.387
	SMV	0.035*	0.305*	0.405*	0.083	0.118	0.155	0.165	0.197	0.241	0.157	0.269	0.377
	NeuralQPP	0.307*	0.296*	0.39*	0.218	0.212	0.274	0.348*	0.349*	0.454*	0.261*	0.241*	0.303*
	NQA-QPP	0.423*	0.359*	0.469*	0.162	0.198	0.245	0.366*	0.337*	0.435*	0.279	0.349*	0.46*
	qppBERT-PL	0.333*	0.274*	0.36*	**0.32***	0.292*	0.36*	0.342*	0.259*	0.326*	0.217	0.15	0.19
	M-QPPF (LSTM)	**0.596***	**0.446***	**0.573***	0.314*	**0.408***	**0.504***	**0.391***	**0.396***	**0.486***	**0.327***	**0.35***	**0.44***
	M-QPPF (GRU)	**0.592***	**0.443***	**0.569***	0.296	**0.408***	**0.494***	**0.381***	**0.371***	**0.454***	**0.368***	**0.408***	**0.514***
k = 20	Clarity	0.001	0.011	0.015	-0.099	-0.153	-0.203	-0.033	-0.005	-0.007	-0.06	-0.075	-0.098
	WIG	0.06*	0.068*	0.091*	0.13	0.102	0.128	0.116	0.092	0.122	0.196	0.234*	0.312*
	NQC	0.045*	0.303*	0.403*	0.121	0.122	0.16	0.119	0.153	0.194	0.192	0.298*	0.398*
	SMV	0.044*	0.298*	0.397*	0.111	0.125	0.16	0.112	0.138	0.181	0.183	0.29*	0.388*
	NeuralQPP	0.296*	0.294*	0.389*	0.22	0.198	0.261	0.333*	0.328*	0.421*	0.266	0.249*	0.307*
	NQA-QPP	0.416*	0.367*	0.48*	0.143	0.161	0.201	0.343*	0.336*	0.425*	0.196	0.298*	0.391*
	qppBERT-PL	0.387*	0.316*	0.415*	**0.444***	0.352*	0.439*	0.351*	0.288*	0.363*	0.369*	0.275*	0.355*
	M-QPPF (LSTM)	**0.61***	**0.472***	**0.601***	0.335*	**0.382***	**0.473***	**0.383***	**0.389***	**0.466***	0.249	**0.318***	**0.406***
	M-QPPF (GRU)	**0.604***	**0.47***	**0.598***	0.368*	**0.398***	**0.486***	**0.402***	**0.419***	**0.513***	**0.394***	**0.413***	**0.502***

Table 7
Comparison between the effectiveness of M-QPPF in the re-ranking task and the baselines.

		MS MARCO Dev set	TREC DL 2019	TREC DL 2020	DL-Hard
		MRR@10	NDCG@10		
k = 10	BM25 (retrieval)	0.187	0.506	0.48	0.304
	BERT Base	**0.279**	0.551	0.516	0.332
	TwinBERT	0.265	0.554	0.511	0.33
	monoT5	**0.279**	0.553	0.523	0.327
	ColBERT	0.275	0.552	0.516	0.331
	PreTTR	0.273	0.553	0.517	0.326
	CoCondenser	0.18	0.498	0.468	0.292
	SPLADE-CoCondenser-EnsembleDistil	0.18	0.498	0.468	0.292
	M-QPPF (LSTM)	0.276	**0.87**	**0.863**	**0.573**
	M-QPPF (GRU)	0.273	**0.875**	**0.866**	**0.577**
k = 20	BM25 (retrieval)	0.187	0.506	0.48	0.304
	BERT Base	0.314	0.636	0.596	0.332
	TwinBERT	0.294	0.634	0.585	0.325
	monoT5	**0.315**	0.634	0.604	0.324
	ColBERT	0.31	0.633	0.601	0.326
	PreTTR	0.304	0.633	0.596	0.329
	CoCondenser	0.172	0.518	0.493	0.262
	SPLADE-CoCondenser-EnsembleDistil	0.172	0.518	0.493	0.262
	M-QPPF (LSTM)	0.304	**0.83**	**0.828**	**0.509**
	M-QPPF (GRU)	0.301	**0.822**	**0.825**	**0.518**

both values of k while it does not show strong performance on the other two datasets. Also, in TREC DL 2019 and DL-Hard datasets, the best baseline changes when the values of k are equal to 10 and 20. For example, TwinBERT and BERT Base are the best baselines for k equal to 10 and 20 respectively in TREC DL 2019.

In the MS MARCO Dev dataset, M-QPPF shows performance improvement compared to all baselines except monoT5 and BERT Base when k is equal to 10. We also observe that the performance of M-QPPF with LSTM is competitive with the best baseline with a slight difference (0.279 vs 0.276). For other values of k, although M-QPPF does not lead to performance improvements compared to the best baseline, namely monoT5, it does however show better performance compared to a number of other baselines such as TwinBERT, CoCondenser, and SPLADE-CoCondenser-EnsembleDistil. However, it is important to note that while monoT5 shows better performance compared to our proposed M-QPPF approach on the MS MARCO Dev dataset (0.279 vs 0.276 at $k = 10$ and 0.315 vs 0.301 at $k = 20$), it shows consistently weaker performance on the other three datasets and for different values of k. For example on the DL-Hard dataset, which consists of difficult queries our method significantly outperforms monoT5 (0.553 vs 0.87 at $k = 10$ and 0.634 vs 0.83 at $k = 20$). The same is true in the other two datasets.

In summary, we find that our proposed M-QPPF approach is able to show strong and robust performance for both ranking and query performance prediction tasks on a wide range of datasets and in comparison with strong baseline methods for each of the tasks. This is an indication that a multitask learning approach has allowed us to benefit from the synergies between the two tasks to improve the performance on each individual task separately.

5.6.4. Ablation study

In this section, we perform an ablation study to explore how learning the model in a multi-task learning setting could lead to improvements over learning each task separately. To understand the importance of the multi-task learning approach in M-QPPF, we consider two variations of M-QPPF where each variation is only optimized for one task, namely query performance prediction (S-QPPF) and ranking (S-RankF). We are interested to see whether the integration of both tasks in M-QPPF through multi-task learning is able to act as effectively as when the models are trained specifically for each individual task. The ideal scenario would be when M-QPPF shows equal or better performance to S-QPPF and S-RankF models on their respective tasks. Tables 8-11 show the evaluation results of these models over the four datasets on both tasks. As expected, the results of S-QPPF and S-RankF are not competitive on the tasks they are not trained on. In other words, S-QPPF performs very poorly on ranking and S-RankF shows weak performance for query performance prediction.

Furthermore, based on the results in these tables and in the context of the QPP task, we observe that M-QPPF significantly outperforms S-QPPF over four evaluation datasets in terms of Pearson ρ, Kendall's τ, and Spearman ρ, which shows the effectiveness of the multi-task objective over the single-task objective for the QPP task. This means that learning the model in a multitask learning setting can bring improved performance when predicting query performance. The main reason is that M-QPPF enables the transfer of shared knowledge from the two related tasks of QPP and ranking. In our proposed multi-task learning framework, the BERT encoding of the query-document pair is shared across the two related tasks, which can be beneficial for improving the QPP task by leveraging information from the document ranking task.

We also empirically investigate whether the model trained with the multitask objective could improve the performance of the ranking task. The results indicate that M-QPPF significantly outperforms S-RankF in terms of retrieval effectiveness except on the MS MARCO Dev set where the performance of M-QPPF is competitive with S-RankF (0.304 vs 0.306 at k equal to 20). This is an

Table 8
Ablation study for performance analysis of M-QPPF on MS MARCO Dev set. * denotes statistically significant on MRR@10 with a two-tailed paired t-test at a 95% confidence interval.

		Pearson ρ	Kendall τ	Spearman ρ	MRR@10
k = 10	S-RankF	-0.226*	-0.204*	-0.252*	0.27
	S-QPPF	0.568*	0.433*	0.556*	0.08
	M-QPPF(LSTM)	**0.596***	**0.446***	**0.573***	**0.276**
k = 20	S-RankF	-0.041*	-0.028*	-0.034*	**0.306**
	S-QPPF	0.338*	0.297*	0.392*	0.047
	M-QPPF(LSTM)	**0.61***	**0.472***	**0.601***	0.304

Table 9
Ablation study for performance analysis of M-QPPF on TREC DL 2019. * denotes statistically significant on MRR@10 with a two-tailed paired t-test at a 95% confidence interval.

		Pearson ρ	Kendall τ	Spearman ρ	NDCG@10
k = 10	S-RankF	0.198	0.13	0.173	0.539
	S-QPPF	**0.363***	0.266*	0.374*	0.430
	M-QPPF(LSTM)	0.314*	**0.408***	**0.504***	**0.87**
k = 20	S-RankF	-0.049	-0.011	-0.008	0.615
	S-QPPF	0.279	0.123	0.175	0.539
	M-QPPF(LSTM)	**0.335***	**0.382***	**0.473***	**0.83**

Table 10
Ablation study for performance analysis of M-QPPF on TREC DL 2020. * denotes statistically significant on MRR@10 with a two-tailed paired t-test at a 95% confidence interval.

		Pearson ρ	Kendall τ	Spearman ρ	NDCG@10
k = 10	S-RankF	0.089	0.066	0.075	0.512
	S-QPPF	0.357*	0.271*	0.389*	0.397
	M-QPPF(LSTM)	**0.391***	**0.396***	**0.486***	**0.863**
k = 20	S-RankF	0.250	0.145	0.205	0.587
	S-QPPF	0.293	0.173	0.241	0.414
	M-QPPF(LSTM)	**0.383***	**0.389***	**0.466***	**0.828**

Table 11
Ablation study for performance analysis of M-QPPF on DL-Hard. * denotes statistically significant on MRR@10 with a two-tailed paired t-test at a 95% confidence interval.

		Pearson ρ	Kendall τ	Spearman ρ	NDCG@10
k = 10	S-RankF	-0.231	-0.185	0.240	0.328
	S-QPPF	0.223	0.283*	0.43*	0.233
	M-QPPF(LSTM)	**0.327***	**0.35***	**0.44***	**0.573**
k = 20	S-RankF	0.056	0.065	0.082	0.313
	S-QPPF	**0.269**	0.264	0.382*	0.236
	M-QPPF(LSTM)	0.249	**0.318***	**0.406***	**0.509**

indication that M-QPPF is not only able to perform query performance prediction better than a model specifically trained only for the QPP task, but is also able to show strong performance on the ranking task.

Finally, we investigate the importance of an appropriate choice of weighting between each loss. Based on Equation (3), the same weights are assigned to ℓ^q_{rank} and ℓ^q_{QPP} in M-QPPF, which is a general approach where all losses are simply summed with an equal contribution. We use $\ell^q = \lambda\ell^q_{rank} + (1-\lambda)\ell^q_{QPP}$ to analyze the combined weights of two different loss functions for $\lambda \in [0,1]$. The performance of M-QPPF with LSTM over MS MARCO Dev set, when k was set to 10, is reported in Table 12 for different values of λ. Overall, we observed that learning with multiple losses improves both document ranking and query performance prediction tasks. In fact, we observe up to 0.3% and 0.4% (i.e. $\lambda = \{0.6, 0.8\}$) increase in the performance of document ranking when using a combination of $l\ell^q_{rank}$ and ℓ^q_{QPP} compared to S-RankF where ℓ^q_{rank} is only used (i.e. $\lambda = 1$). Similarly, for QPP task, we observe that the performance improves up to 4.3% and 4.5% (i.e., $\lambda = \{0.4, 0.6\}$) in terms of Pearson ρ when combining both losses compared to using ℓ^q_{QPP} only.

Table 12
The performance of M-QPPF with the combined weights of two different loss functions.

λ	Pearson ρ	Kendall's τ	Spearman ρ	MRR@10
0.0 (S-QPPF)	0.568*	0.433*	0.556*	0.08
0.1	0.603*	0.448*	0.573*	0.265
0.2	0.575*	0.452*	0.580*	0.256
0.3	0.606*	0.447*	0.572*	0.269
0.4	0.611*	0.448*	0.574*	0.269
0.5	0.610*	0.445*	0.569*	0.272
0.6	0.613*	0.446*	0.570*	0.273
0.7	0.610*	0.443*	0.567*	0.272
0.8	0.600*	0.436*	0.559*	0.274
0.9	0.595*	0.425*	0.547*	0.270
1.0 (S-RankF)	-0.226*	-0.204*	-0.252*	0.27

5.6.5. Analysis and discussion

Beyond the information retrieval and correlation metrics that report the performance of our proposed M-QPPF approach, it is also important to understand the details of M-QPPF when succeeding in predicting the performance of the queries and ranking the retrieved documents. To this end, we select three examples and explain the details of each of them in the context of our work.

As the first example, the queries "vitamin e anti scar" with $NDCG@10$ of 0.0732 and "why does lacquered brass tarnish" with $NDCG@10$ of 0.3988 are selected from the DL-Hard dataset. The performance of the first query is near zero and lower than the second one which means that the first query is more difficult than the second for the BM25 ranking method. The predicted performance of the second query is higher than the first by M-QPPF due to the fact that M-QPPF is trained to be able to predict the query performance near the actual performance. In other words, the first query is more difficult than the second one in view of M-QPPF, unlike NQA-QPP as the best baseline, which estimates that the second query (with the predicted performance of 0.1376) is more difficult than the first one (with the predicted performance of 0.2064). The main reason for this is that M-QPPF benefits from the multi-task learning framework and utilizes the knowledge provided by the ranking task. To understand this, we referred to our model which trained under a single-task framework and learned to predict the query performance (S-QPPF). S-QPPF is similar to NQA-QPP and predicted the performance of the first query to be higher than the second. This example shows how a multi-task learning approach leads to better performance on specific queries.

Also, we believe that knowledge encoded by the QPP task affects the ranking task. For the two queries mentioned in the previous paragraph, one can observe an improvement of 6.28% by BERT Base which is the best baseline in the ranking task while M-QPPF is able to improve performance by up to 18%. We would like to point out that the architecture of M-QPPF in the ranking task is quite similar to the BERT-base baseline in which query and document input sequences are concatenated with special tokens and then the [CLS] vector is computed by BERT and fed into a single-layer neural network. In the above example, our proposed method is able to show improvements in comparison to the baseline in the ranking task due to the fact that it benefited from the QPP knowledge. Because of this, the predicted ranking performance by S-RankF, which is trained under a single-task framework for predicting the relevance score of documents, is close to that of BERT-base whilst our multi-tasking approach, M-QPPF, shows significantly better performance.

As another example, we selected two queries "how big is magic kingdom in acres" and "what is a photostat" from the MS MARCO Dev set. The performance of the first and second queries is 0.1429 and 0.1667, respectively in terms of $MRR@10$ when BM25 method is used to retrieve the documents. The difference between the actual performances of the two queries is small and about 2.4%. Since M-QPPF is trained based on the actual performance of BM25, it is able to predict the performance of the first query to be less than the second with a difference of 0.37%. NQA-QPP, as the best BERT-based baseline, estimated the second query to be more difficult than the first with a high difference of -11.61%. Our method outperforms the state-of-the-art baseline because it integrates both tasks of QPP and ranking through multi-task learning and employs their shared knowledge. In order to better understand the effect of this on the ranking task, we consider the predicted performance by S-QPPF which is only optimized for that QPP task. S-QPPF is similar to NQA-QPP and predicted the performance of the first query to be higher than the second query with a difference of 3.93%. We can even see the effect of multi-task learning on the ranking task when S-RankF was not able to improve the ranking performance of the second query while M-QPPF improved it by 3.33%. Like S-RankF, BERT Base leads to no change in the ranking performance on the second query.

As the last example to show the strength of multi-task learning, we consider two queries "white pine millings in wv" and "what is the empirical formula for phosphorus selenide" with the actual performance of 0.2 and 0.25, respectively, from the MS MARCO Dev set. Our proposed M-QPPF approach is able to perform better against the state-of-the-art baselines on both tasks and even two versions of the S-QPPF and S-RankF that are trained specifically for each task of QPP and ranking, respectively. The predicted performance of the first query is lower than the second with a difference of 3.59% by M-QPPF, which means that the first query is harder than the second in terms of our model similar to BM25. In addition to NQA-QPP that failed to estimate that the first query is harder than the second with a difference of -5.77%, qppBERT-PL, as another state-of-the-art method, could not predict the performance of both queries (a high difference of -17.06%). S-QPPF, as the version of our model trained for the individual task of QPP, leads to an incorrect estimation about these two queries by predicting the performance of the first and second queries at 0.412 and 0.3367, respectively. This confirms that learning the model in a multi-task learning setting can bring improved performance when predicting

Table 13
Exploratory examples contrasting the performance of our proposed approach with the state-of-the-art baselines.

	ID	Text	Actual performance	Query performance prediction			Ranking performance		
			BM25	M-QPPF	NQA-QPP	S-QPPF	M-QPPF	BERT Base	S-RankF
Example 1	537817	"vitamin e anti scar"	$NDCG@10$: 0.0732	0.1335	0.2064	0.3365	0.0906	0.0906	0.0979
	1136769	"why does lacquered brass tarnish"	$NDCG@10$: 0.3988	0.2146	0.1376	0.0295	0.5792	0.4614	0.5583
Example 2	1099656	"how big is magic kingdom in acres"	$MRR@10$: 0.1429	0.0709	0.2849	0.1613	0.3333	0.1667	0.25
	694678	"what is a photostat"	$MRR@10$: 0.1667	0.0746	0.1688	0.122	0.2	0.1667	0.1667
Example 3	992433	"white pine millings in wv"	$MRR@10$: 0.2	0.452	0.4325	0.4121	1.0	0.25	0.5
	1040038	"what is the empirical formula for phosphorus selenide	$MRR@10$: 0.25	0.4879	0.3746	0.3367	0.3333	0.25	0.25

query performance. In the ranking task, we have an improvement of 80% and 8.33% by M-QPPF for the first and second queries, respectively. In contrast, there is no improvement in the ranking performance for the second query by BERT-base and S-RankF, however, they were able to produce 5% and 30% improvements for the first query. Given that the M-QPPF's architecture in the ranking task is similar to BERT-base, and the similarity between BERT-base and S-RankF in terms of performance, we can conclude that M-QPPF in the context of the ranking task is highly positively influenced by the QPP task. The details of these examples are shown in Table 13.

5.6.6. Time complexity analysis

In this subsection, we study the time complexity of M-QPPF at inference time. In terms of each task, M-QPPF is a sequential model where the query-document pairs are first encoded by BERT, and then their representations are fed into the task-specific layers. So, the time complexity of the proposed model is additive by the BERT and the task-specific layers. The total time complexity of M-QPPF for the QPP task can be computed as followed:

$$time_{QPP} \approx O(k \times n^2 \times d) + O(k \times d^2) + O(d \times h) + O(h) \tag{18}$$

where $O(k \times n^2 \times d)$ is the time complexity of BERT. The time complexity of BERT is mainly concentrated in the Self-Attention module. Here, k is the number of documents in the retrieved list, d is the embedding dimension, and n is the total words in the sequence of the query-document pair. Given that BERT computes attention weights for each word with respect to every other word, $O(n)$ operations for each word and therefore $O(n^2)$ for all the words are performed. $O(k \times d^2)$ is the time complexity of RNN for both LSTM and GRU. In RNN, matrix multiplication of the hidden states of the previous steps takes d^2 operations. For k-steps to be processed, $O(k \times d^2)$ operations are needed. The time complexity of the first and second fully connected layers is $O(d \times h)$ and $O(h)$ where h is the number of neurons.

The total time complexity of M-QPPF for the ranking task would be as follows:

$$time_{Rank} \approx O(k \times n^2 \times d) + O(k \times d) \tag{19}$$

Similar to the QPP task, $O(k \times n^2 \times d)$ is the time complexity of BERT which is shared for both tasks, and $O(k \times d)$ is the time complexity of rank-specific layers that is a simple fully connected layer with d inputs and one output that is $O(d \times 1)$. For k-query-document pairs to be processed, $O(k \times d)$ operations would be needed.

6. Concluding remarks

In this work, we have proposed M-QPPF, a multi-task learning framework to jointly learn the two main tasks of query performance prediction and ad hoc retrieval. M-QPPF relies on the fine-tuning of BERT in order to learn to predict the quality of the ranked list of retrieved documents for a given input query and re-rank that list simultaneously.

Our proposed framework has been evaluated by performing a series of experiments over four sets of queries from MS MARCO. The queries ranking by M-QPPF and their actual performance based on true MRR@10 have been compared using three correlation metrics to evaluate the performance of M-QPPF for the QPP task. Also, the performance of M-QPPF on the ranking task against the BERT-based baselines for ad hoc retrieval has been compared.

We would like to extend the work in this paper in the following directions:

- Although M-QPPF is more effective at determining the hard queries against the state-of-the-art, it does not currently specifically focus on improving the performance of hard queries. Hard queries can often be due to issues such as vocabulary mismatch, and hence they could be improved through query reformulation. For this reason, as future work, we are interested in exploring the impact of the multi-task learning framework on reformulating hard queries so that the framework learns to predict query performance and also reformulates the identified hard queries at the same time;

- M-QPPF as a post-retrieval model considers both the ranked list of the retrieved documents and the queries to predict the performance of the queries. This is unlike pre-retrieval methods that rely solely on the queries themselves and the information contained in them. Given that pre-retrieval predictors [19] show promise in improving query performance prediction, we are interested in exploring their potential role under a multi-task learning scenario.

CRediT authorship contribution statement

Maryam Khodabakhsh: Conceptualization, Formal analysis, Investigation, Software, Validation, Writing – original draft. **Ebrahim Bagheri:** Conceptualization, Investigation, Methodology, Writing – review & editing.

Declaration of competing interest

We have no conflict of interest to declare.

Data availability

Data are available.

References

[1] T. Formal, C. Lassance, B. Piwowarski, S. Clinchant, From distillation to hard negative sampling: making sparse neural IR models more effective, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2353–2359.
[2] O. Khattab, M. Zaharia, ColBERT: efficient and effective passage search via contextualized late interaction over BERT, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 39–48.
[3] R. Nogueira, Z. Jiang, R. Pradeep, J. Lin, Document ranking with a pretrained sequence-to-sequence model, in: Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.
[4] T.A. Nakamura, P.H. Calais, D. de Castro Reis, A.P. Lemos, An anatomy for neural search engines, Inf. Sci. 480 (2019) 339–353.
[5] J. Lin, R. Nogueira, A. Yates, Pretrained transformers for text ranking: BERT and beyond, Synth. Lect. Hum. Lang. Technol. 14 (4) (2021) 1–325.
[6] E. Bagheri, F. Al-Obeidat, A latent model for ad hoc table retrieval, in: European Conference on Information Retrieval, Springer, 2020, pp. 86–93.
[7] D. Carmel, E. Yom-Tov, Estimating the query difficulty for information retrieval, Synth. Lect. Inf. Concept. Retr. Services 2 (1) (2010) 1–89.
[8] S. Cronen-Townsend, Y. Zhou, W.B. Croft, Predicting query performance, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002, pp. 299–306.
[9] Y. Zhou, W.B. Croft, Query performance prediction in web search environments, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007, pp. 543–550.
[10] A. Shtok, O. Kurland, D. Carmel, F. Raiber, G. Markovits, Predicting query performance by query-drift estimation, ACM Trans. Inf. Syst. 30 (2) (2012) 1–35.
[11] Y. Tao, S. Wu, Query performance prediction by considering score magnitude and variance together, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 1891–1894.
[12] H. Zamani, W.B. Croft, J.S. Culpepper, Neural query performance prediction using weak supervision from multiple signals, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 105–114.
[13] H. Hashemi, H. Zamani, W.B. Croft, Performance prediction for non-factoid question answering, in: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, 2019, pp. 55–58.
[14] S. Datta, S. MacAvaney, D. Ganguly, D. Greene, A 'pointwise-query, listwise-document' based qpp approach, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022.
[15] X. Chen, B. He, L. Sun, Groupwise query performance prediction with BERT, in: European Conference on Information Retrieval, Springer, 2022, pp. 64–74.
[16] H. Roitman, S. Erera, G. Feigenblat, A study of query performance prediction for answer quality determination, in: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, 2019, pp. 43–46.
[17] S. Sarnikar, Z. Zhang, J.L. Zhao, Query-performance prediction for effective query routing in domain-specific repositories, J. Assoc. Inf. Sci. Technol. 65 (8) (2014) 1597–1614.
[18] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[19] N. Arabzadeh, F. Zarrinkalam, J. Jovanovic, E. Bagheri, Neural embedding-based metrics for pre-retrieval query performance prediction, in: European Conference on Information Retrieval, Springer, 2020, pp. 78–85.
[20] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186.
[21] B. He, I. Ounis, Inferring query performance using pre-retrieval predictors, in: International Symposium on String Processing and Information Retrieval, Springer, 2004, pp. 43–54.
[22] Y. Zhou, W.B. Croft, Ranking robustness: a novel framework to predict query performance, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, 2006, pp. 567–574.
[23] J.A. Aslam, V. Pavlu, Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions, in: European Conference on Information Retrieval, Springer, 2007, pp. 198–209.
[24] M. Khodabakhsh, E. Bagheri, Semantics-enabled query performance prediction for ad hoc table retrieval, Inf. Process. Manag. 58 (1) (2021) 102399.
[25] N. Arabzadeh, M. Khodabakhsh, E. Bagheri, Bert-qpp: contextualized pre-trained transformers for query performance prediction, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 2857–2861.
[26] E. Bagheri, F. Ensan, F.N. Al-Obeidat, Neural word and entity embeddings for ad hoc retrieval, Inf. Process. Manag. 54 (4) (2018) 657–673.
[27] M. Khodabakhsh, E. Bagheri, Qualitative measures for ad hoc table retrieval, Inf. Sci. 607 (2022) 1–26.
[28] R. Nogueira, K. Cho, Passage re-ranking with BERT, arXiv preprint, arXiv:1901.04085, 2019.
[29] S. MacAvaney, A. Yates, A. Cohan, N. Goharian, Cedr: contextualized embeddings for document ranking, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1101–1104.
[30] Z.A. Yilmaz, W. Yang, H. Zhang, J. Lin, Cross-domain modeling of sentence-level evidence for document retrieval, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3490–3496.

[31] W. Lu, J. Jiao, R. Zhang, TwinBERT: distilling knowledge to twin-structured compressed BERT models for large-scale retrieval, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 2645–2652.
[32] Y. Luan, J. Eisenstein, K. Toutanova, M. Collins, Sparse, dense, and attentional representations for text retrieval, Trans. Assoc. Comput. Linguist. 9 (2021) 329–345.
[33] S. MacAvaney, F.M. Nardini, R. Perego, N. Tonellotto, N. Goharian, O. Frieder, Efficient document re-ranking for transformers by precomputing term representations, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 49–58.
[34] J. Lin, R. Nogueira, A. Yates, Pretrained Transformers for Text Ranking: BERT and Beyond, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2021.
[35] Y. Zhang, Q. Yang, A survey on multi-task learning, IEEE Trans. Knowl. Data Eng. 34 (2021) 5586–5609.
[36] X. Liu, J. Gao, X. He, L. Deng, K. Duh, Y.-y. Wang, Representation learning using multi-task deep neural networks for semantic classification and information retrieval, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, Association for Computational Linguistics, May–June 2015, pp. 912–921.
[37] W.U. Ahmad, K.-W. Chang, Multi-task learning for document ranking and query suggestion, in: Sixth International Conference on Learning Representations, 2018.
[38] B. Liu, H. Zamani, X. Lu, J.S. Culpepper, Generalizing discriminative retrieval models using generative tasks, in: Proceedings of the Web Conference 2021, 2021, pp. 3745–3756.
[39] Q. Cheng, Z. Ren, Y. Lin, P. Ren, Z. Chen, X. Liu, M.d. de Rijke, Long short-term session search: joint personalized reranking and next query prediction, in: Proceedings of the Web Conference 2021, 2021, pp. 239–248.
[40] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to rank: from pairwise approach to listwise approach, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 129–136.
[41] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[42] A. Graves, Long short-term memory, in: Supervised Sequence Labelling with Recurrent Neural Networks, 2012, pp. 37–45.
[43] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches, in: Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014.
[44] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, Ms marco: a human generated machine reading comprehension dataset, in: CoCo@ NIPs, 2016.
[45] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E.M. Voorhees, Overview of the TREC 2019 deep learning track, in: Text REtrieval Conference (TREC), 2020.
[46] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, Overview of the TREC 2020 deep learning track, CoRR, arXiv:2102.07662, 2021.
[47] I. Mackie, J. Dalton, A. Yates, How deep is your learning: the dl-hard annotated deep learning dataset, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 2335–2341.
[48] L. Gao, J. Callan, Unsupervised corpus aware language model pre-training for dense passage retrieval, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, Association for Computational Linguistics, May 2022, pp. 2843–2853.
[49] T. Formal, B. Piwowarski, S. Clinchant, Splade: sparse lexical and expansion model for first stage ranking, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 2288–2292.
[50] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in: NIPS 2014 Workshop on Deep Learning, December 2014, 2014.