Neural Query Performance Prediction using Weak Supervision from Multiple Signals

Hamed Zamani
University of Massachusetts Amherst
Amherst, MA 01003
zamani@cs.umass.edu

W. Bruce Croft
University of Massachusetts Amherst
Amherst, MA 01003
croft@cs.umass.edu

J. Shane Culpepper
RMIT University
Melbourne, Australia
shane.culpepper@rmit.edu.au

ABSTRACT

Predicting the performance of a search engine for a given query is a fundamental and challenging task in information retrieval. Accurate performance predictors can be used in various ways, such as triggering an action, choosing the most effective ranking function per query, or selecting the best variant from multiple query formulations. In this paper, we propose a *general* end-to-end query performance prediction framework based on neural networks, called *NeuralQPP*. Our framework consists of multiple components, each learning a representation suitable for performance prediction. These representations are then aggregated and fed into a prediction subnetwork. We train our models with multiple weak supervision signals, which is an unsupervised learning approach that uses the existing unsupervised performance predictors using weak labels. We also propose a simple yet effective component dropout technique to regularize our model. Our experiments on four newswire and web collections demonstrate that NeuralQPP significantly outperforms state-of-the-art baselines, in nearly every case. Furthermore, we thoroughly analyze the effectiveness of each component, each weak supervision signal, and all resulting combinations in our experiments.

ACM Reference Format:
Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. 2018. Neural Query Performance Prediction using Weak Supervision from Multiple Signals. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210041

1 INTRODUCTION

Query performance prediction (QPP) is a well studied problem in information retrieval (IR) due to its potential importance in improving the effectiveness and efficiency of a wide variety of search tasks [5]. The query performance prediction task is defined as predicting the quality of a retrieval model for a given query, when neither explicit nor implicit relevance information is available. Accurate and real-time performance predictors could potentially be used in triggering a specific action in the retrieval system, such as selecting an index traversal algorithm at query time [27], choosing the correct number of documents to process in a cascaded multistage retrieval system [13], choosing the most effective ranking function per query, selecting the best variant from multiple query reformulations, or requesting more information from users in cases of potential poor retrieval performance, particularly in conversational systems. Query performance prediction models are categorized as pre-retrieval and post-retrieval approaches. Post-retrieval approaches, which are the focus of this paper, analyze the result list returned by the retrieval engine in response to the query. Post-retrieval predictors are our focus as they have been proven to be more effective than pre-retrieval predictors [5].

In this paper, we propose a *general* framework based on neural networks for the query performance prediction task. Our framework, called NeuralQPP, consists of multiple components, each analyzing a distinct aspect useful for performance prediction. Each component learns a high-dimensional dense representation suitable for the QPP task. These representations are then aggregated and fed into a prediction sub-network. The whole framework is trained in an end-to-end fashion.

We introduce three neural components for implementing the NeuralQPP framework. Each is designed with minimal network engineering for simplicity. The first component analyzes the retrieval scores for the top documents retrieved in response to a given query. The retrieval score distribution has been previously used in a variety of QPP models [14, 40, 44, 57]. The second component analyzes the term distribution for the documents appearing in the result list. A term distribution can be a means for measuring the coherence of the top ranked documents, which has been proven to be highly correlated with query performance [11]. The third component analyzes the distributed representation of documents in a semantic space. This component is able to measure the semantic coherence and diversity of the result list.

Recently, Dehghani et al. [16] and Zamani and Croft [50] proposed the training of Neural IR models with weak supervision. Weak supervision is an *unsupervised* learning approach where a large set of unlabeled data is labeled with an existing unsupervised model as a weak labeler. As it is often very difficult to generate high quality training data, we describe an approach to training NeuralQPP using *multiple* weak supervision signals. To do so, we are able to benefit from three existing predictors that estimate the query performance based on different intuitions and assumptions. To be exact, our weak labelers include a clarity-based approach by Cronen-Townsend et al. [11], a score-based approach by Shtok et al. [44], and a combining approach by Shtok et al. [42]. Training a generalized model with multiple weak signals led us to develop a component dropout technique that randomly disables at most $K-1$ (out of K) components of the NeuralQPP framework. This can

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210041

be also viewed as a regularization that prevents the models from overfitting.

We evaluate our models using four standard TREC collections, including two newswire test collections (AP and Robust) respectively used for the TREC 1-3 Ad-hoc Tracks and the TREC 2004 Robust Track, and two large-scale web collections (GOV2 and ClueWeb) respectively used for the TREC 2004-2006 Terabyte Tracks and the TREC 2009-2012 Web Tracks. Our experiments show that the proposed model significantly outperforms the baselines, in nearly every setting. We also empirically study the influence of each component in the NeuralQPP framework, and the effectiveness of employing multiple weak signals for training. The results demonstrate that NeuralQPP performs remarkably well in predicting the performance of various retrieval models.

In summary, this paper introduces the first neural network architecture for query performance prediction. Furthermore, it not only provides a successful implementation of the weak supervision idea for an additional fundamental IR task, but also provides new insights on how best to learn with multiple weak labelers. NeuralQPP produces state-of-the-art performance on multiple collections.

2 RELATED WORK

In this section, we first present previous work on query performance prediction, and next briefly review the literature on weak supervision for information retrieval.

2.1 Query Performance Prediction

Quality estimation is a fundamental task that can help to improve effectiveness or efficiency in various applications, such as machine translation [45], and automatic speech recognition [4, 31]. When it comes to search engines, the task is called query performance or query difficulty prediction. This task has been widely studied in the IR literature [5, 11, 12, 14, 18, 22, 42–44, 47, 56, 57]. The task of query performance prediction is defined as predicting the retrieval effectiveness of a search engine given an issued query with no implicit or explicit relevance information.

Query performance prediction approaches can be partitioned into two disjoint sets: pre-retrieval and post-retrieval approaches. Pre-retrieval QPP approaches predict the performance of each query based on the content and the context of the query in addition to the corpus statistics. Pre-retrieval predictors are often derived from linguistic or statistical information. Part-of-speech tags, as well as syntactic and morphological features of query terms are among the linguistic features used for query performance prediction [30]. Inverse document frequency [11] and average query term coherence [19] are examples of statistical information used for this task. Hauff et al. [18] provided a through overview of the pre-retrieval QPP approaches.

Alternately, post-retrieval QPP approaches, which are the focus of this paper, estimate the query performance by analyzing the result list returned by the retrieval engine in response to the query. Carmel and Yom-Tov [5] categorized post-retrieval predictors as clarity-based, robustness-based, and score-based approaches:

- Clarity-based approaches [11, 12] estimate the query performance by measuring the coherence (clarity) of the result list with respect to the collection. These approaches assume that the more focused the result list, the more effective the retrieval.
- Robustness-based approaches predict the query performance by estimating the *robustness* of the result list. Robustness can be measured in various ways. For example, Zhou and Croft [57] measured it based on query perturbation in a Query Feedback (QF) model. In other work, the same authors [56] measured the ranking robustness through document perturbation by injecting noise into the top results. Both query and document perturbations were also studied by Vinay et al. [49]. Aslam and Pavlu [2] studied the ranking robustness based on retrieval engine perturbation. Apart from perturbation approaches, Diaz [17] measured the ranking robustness using the cluster hypothesis [48] by regularizing the retrieval score of each document given its most similar documents. This approach is called spatial autocorrelation.
- A variety of post-retrieval approaches predict the query performance by analyzing the retrieval score distribution, and are commonly referred to as score-based approaches. Among these, the Weighted Information Gain (WIG) of Zhou and Croft [57] and the Normalized Query Commitment (NQC) of Shtok et al. [44] are the most popular QPP models, and are considered state-of-the-art. WIG measures the divergence of the mean retrieval score from the collection score and NQC measures the standard deviation of the retrieval scores normalized by the collection score. Retrieval score distribution has been further employed in other models [14, 34] for the QPP task. Most recently, Roitman et al. [40] proposed a bootstrapping approach to provide a robust standard deviation estimator for retrieval scores.

There is also a line of research that combines multiple predictors from multiple categories. The utility estimation framework (UEF) of Shtok et al. [42] is an example of this QPP family, which is based on statistical decision theory. Making use of both pseudo-effective and pseudo-ineffective reference lists was further studied by Kurland et al. [22], Shtok et al. [43], and Roitman [38].

There also exist a set of supervised approaches for query performance prediction. For instance, Raiber and Kurland [36] proposed a learning to rank model based on Markov random fields and observed significant improvements. Most recently, Roitman et al. [39] introduced a supervised combining approach based on coordinate ascent. Our model does not require human-labeled data for training, and thus supervised approaches are outside the scope of this paper.

2.2 Weak Supervision

Limited training data has been a perennial problem in information retrieval, and many machine learning-related domains [52]. This has motivated researchers to explore building models using *pseudo-labels*. For example, pseudo-relevance feedback (PRF) [9] is one of the long-standing approaches which assumes that the top retrieved documents in response to a given query are relevant to the query, and thus uses these documents to improve the retrieval performance. Although this assumption does not hold in all cases, PRF has been proven to be effective in many retrieval settings [23, 37, 51, 54]. Building pseudo-collections and simulated queries for various IR tasks could be considered as another set of approaches that tackle this issue [1, 3].

As widely known, deep neural networks often require a large volume of training data. Recently, training neural IR models based on pseudo-labels has shown to produce successful results [16, 50]. This learning approach is called *weak supervision*. Dehghani et al. [16] proposed training a neural ranking model for the ad-hoc retrieval task based on the labels generated by an existing retrieval model, such as BM25. Zamani and Croft [50] argued that the objective functions of the general-purpose word embedding models, such as word2vec [29], are not necessarily equivalent to IR objectives. These approaches train relevance-based word embeddings using the output of Lavrenko and Croft's relevance models [23] as a type of weak labeling. Following these studies, the idea of training neural IR models with weak supervision has been further employed [7, 15, 26].

There are two key factors that distinguish our work from previous studies. First, since training neural IR models with weak supervision has been recently shown to be effective, its effectiveness in many common IR tasks remains relatively unexplored. This work introduces a successful implementation of weak supervision for another fundamental IR task. Second, all the previously mentioned studies train models using a single weak label. However, in this work, we describe how to train our NeuralQPP model using multiple weak labels. Learning from multiple weak labelers should lead to higher generalization of the learned models.

3 NEURAL PERFORMANCE PREDICTION

In this section, we propose a *general* query performance prediction framework based on neural networks. The framework is called *NeuralQPP* and is independent of the retrieval engine. NeuralQPP consists of K components that cover different and complementary aspects of query performance prediction. Each component c_i is a sub-network in NeuralQPP, that produces a d_i-dimensional real-valued dense representation, denoted as ρ_i. The learned representations are expected to provide useful information for the query performance prediction task. The obtained ρ_is are then aggregated using an aggregation function Λ which outputs a d-dimensional dense vector. This vector is finally fed into a prediction function Γ that returns a real number representing the predicted performance. All the sub-networks in the NeuralQPP framework are trained simultaneously in an end-to-end fashion.

In summary, the performance of each query is predicted as follows:

$$\Gamma(\Lambda(\rho_1, \rho_2, \cdots, \rho_K)) \tag{1}$$

where ρ_i is the output of the component c_i.

In order to minimize the number of hyper-parameters and perform minimal network engineering, we implement the aggregation function Λ as a weighted average:

$$\Lambda(\rho_1, \rho_2, \cdots, \rho_K) = \frac{1}{K} \sum_{i=1}^{K} \omega_i \rho_i \tag{2}$$

where ω_i controls the influence of each component in the final prediction. The network parameters ω_i are trained as part of the NeuralQPP model. Note that this definition of Λ forces the dimensions of all ρ_is to be equal.

We model the function Γ as a fully connected feed-forward neural network that takes the output of Λ and produces a real number representing the predicted performance. We use rectified linear unit (ReLU) as our activation function for hidden layers to learn non-linear functions, and sigmoid for the output layer. To prevent overfitting, we use dropout in all hidden layers. The number of hidden layers and their sizes are hyper-parameters of the model.

In this paper, we implement three components for NeuralQPP ($K = 3$). The first component analyzes the retrieval score distribution, while the second component considers the term distributions in pseudo-effective and pseudo-ineffective document sets. The last component analyzes the semantic information obtained from the top retrieved documents. The following subsections describe these components, in detail.

Retrieval Scores Analyzer. Inspired by the score-based approaches described in Section 2.1, such as WIG [57] and NQC [44], our first component, called the retrieval scores analyzer, estimates the query performance given the retrieval scores for the top n documents returned by the search engine in response to a query q. As shown in Figure 1a, this component takes a vector $\vec{s}$ with $n + 1$ dimensions as input, such that:

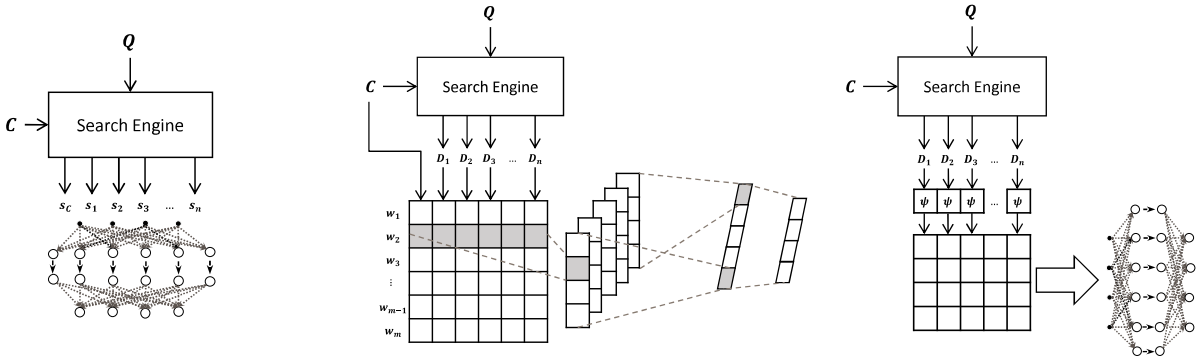
$$s_i = \begin{cases} \text{score}(q, C) & \text{if } i = 1 \\ \text{score}(q, D_{i-1}) & o.w. \end{cases} \tag{3}$$

where C and D_{i-1} denote the collection and the $(i-1)^{\text{th}}$ document in the result list returned by the search engine. 'score' denotes the scoring function used by the retrieval engine and $\text{score}(q, C)$ is computed as the retrieval score for a document constructed by concatenating all documents in the collection. The order of concatenation does not matter for bag of words models. We feed the constructed vector $\vec{s}$ into a fully-connected feed-forward neural network. In summary, this component computes a non-linear abstract representation of the retrieval score distribution, suitable for the query performance prediction task.

Term Distribution Analyzer. Inspired by the clarity-based approaches [11] described in Section 2.1, a term distribution analyzer component predicts the query performance using term distribution information. The component's architecture is presented in Figure 1b. In this component, we first create a matrix $A = [a_{ij}]$ with $n + 1$ columns where the first column corresponds to the collection (as a pseudo-ineffective document set) and each of the remaining n columns corresponds to each of the top n documents retrieved in response to the query q (as pseudo-effective documents). The matrix A has m rows, each corresponding to a vocabulary term from a set W containing the top m terms with the highest cumulative count in the top n retrieved documents ($|W| = m$). Each element of the matrix A is calculated as:

$$a_{ij} = \begin{cases} \Pr(w_i|\theta_C) & \text{if } j = 1 \\ \Pr(w_i|\theta_{D_{j-1}}) & o.w. \end{cases} \tag{4}$$

where w_i, θ_C, and $\theta_{D_{j-1}}$ respectively denote the i^{th} term in the vocabulary set W, the collection's unigram language model, and the unigram language model of the $(i-1)^{\text{th}}$ retrieved document. The language models are estimated using maximum likelihood estimation. Since this component is responsible for term distribution analysis, we can assume that vocabulary terms are independent. Therefore a non-linear mapping function $\phi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^f$ is applied on each row of the matrix A. The parameters of this mapping



(a) Retrieval Scores Analyzer (b) Term Distribution Analyzer (c) Semantic Analyzer

Figure 1: NeuralQPP consists of the three components depicted above. The representations learned by each of these components are then aggregated using the arithmetic mean and then fed into a fully-connected feed-forward network that produces a single score for query performance prediction.

function are shared for all m terms (rows of the matrix). Indeed, this is similar to applying a convolutional layer with the window size and stride of 1. The input channel size and the filter size are equal to m and f, respectively. Therefore, this layer outputs a $f \times m$ matrix. A sub-sampling phase is further applied. We take the maximum value of the f features learned for each term (max-pooling), which results in a m-dimensional vector. This vector is then fed to a fully-connected feed-forward network for dimension reduction and learning an abstract representation of term distributions, expected to be suitable for query performance prediction.

Semantic Analyzer. The semantic analyzer component, shown in Figure 1c, takes the documents returned by the retrieval engine and measures the query performance based on their distributed representations. For instance, this component can measure how semantically coherent or diverse the returned documents are. The intuition behind this is that coherence and diversity in the returned documents correlate with the ambiguity of the query, since a query may carry multiple meanings or intents. Previous QPP models that analyze the coherence of the result list, e.g., clarity [11], are based on term occurrence (similar to our term distribution analyzer component). Thus, this component provides a novel way of looking at the problem.

In this component, we first represent each document in a latent semantic space, and then learn a set of latent features based on the learned representations. Our document representation function ψ consists of two major functions: (1) an embedding function $\mathcal{E} : V \rightarrow \mathbb{R}^l$ that maps each term from the vocabulary set V to a l-dimensional embedding space, and (2) a global term weighting function $\mathcal{W} : V \rightarrow \mathbb{R}$ that maps each vocabulary term to a real-valued number showing its global importance. The document representation function ψ represents a document $D = \{w_1, w_2, \cdots, w_{|D|}\}$ as follows:

$$\psi(D; \mathcal{E}, \mathcal{W}) = \sum_{i=1}^{|D|} \widehat{\mathcal{W}}(w_i) \cdot \mathcal{E}(w_i) \tag{5}$$

which is the weighted element-wise summation over the term embedding vectors. A normalized weight $\widehat{\mathcal{W}}$ is learned for each term using a softmax function as follows:

$$\widehat{\mathcal{W}}(w_i) = \frac{\exp(\mathcal{W}(w_i))}{\sum_{j=1}^{|D|} \exp(\mathcal{W}(w_j))} \tag{6}$$

This approach of document representation is based on the bag of words assumption. Despite its simplicity, it was shown to perform well for ad-hoc retrieval tasks [16].

We flatten, concatenate, and feed the representations learned for the top n retrieved documents ($\{\psi(D_1), \psi(D_2), \cdots, \psi(D_n)\}$) into a fully-connected feed-forward network in order to obtain a non-linear abstract representation that demonstrates useful information for query performance prediction extracted from semantic representation of documents in the result list.

4 TRAINING WITH MULTIPLE WEAK SUPERVISION SIGNALS

In this section, we describe how to train the proposed neural query performance prediction model with no labeled training data. Indeed, we first propose to train our model using multiple weak supervision signals in Section 4.1, and later propose a component dropout technique to regularize our model in Section 4.2. Finally, Section 4.3 introduces the weak supervision signals employed to train the NeuralQPP model.

4.1 Training

Let $\mathcal{M}$ be a retrieval model that retrieves documents from the collection C in response to a given query. In this work, we propose to train a model with multiple weak supervision signals, which is categorized as an unsupervised learning approach. To do so, we first obtain a set of queries Q and N weak labelers: N unsupervised query performance prediction models that can provide us complementary information. Predicting the performance of each query $q_i \in Q$ over the collection C using the weak supervision signals results in a training set $T = \{(q_i, \pi^n_{\mathcal{M}}(q_i; C), \mathcal{Y}_i) : q_i \in Q\}$ where $\pi^n_{\mathcal{M}}(q_i; C)$ denotes the list of the top n documents retrieved by $\mathcal{M}$ in response to the query q_i, and $\mathcal{Y}_i$ denotes a list of predicted performances for q_i by each of the weak supervision signals (thus, $|\mathcal{Y}_i| = N$).

A straightforward solution for learning from multiple weak labels would be casting the problem to learning from a single weak label by aggregating the N weak labels to end up with a single label for each query. This aggregation can be done, for example, by averaging the labels for pointwise settings, or by majority voting for pairwise settings.

Another simple solution would be training N distinct models each using one of the weak labels and then aggregating their outputs at inference time by summation.

In this paper, we argue that neither of these solutions are optimal, since they both incur information loss (which is also justified in our experiments). Therefore, we aim to train our model by optimizing across all weak labels at the same time. Our proposed solution simultaneously optimizes N loss functions, each corresponding to a weak label. Hence, we define our loss function as a linear interpolation of N loss functions:

$$\mathcal{L} = \sum_{k=1}^{N} \alpha_k \mathcal{L}_k \tag{7}$$

where $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_N]$ is a vector of hyper-parameters controlling the influence of each weak label in the final loss function. We investigate two learning settings in our experiments: pointwise and pairwise.

Pointwise learning. In a pointwise setting, we use mean absolute error (MAE) as the loss function.[1] The absolute error for a query q_i in the training set is defined as follows:

$$\mathcal{L}_k(q_i) = |\mathcal{Y}_{ik} - \widehat{P}_k(q_i; \mathcal{M}, C, \theta)| \tag{8}$$

where $\widehat{P}$ denotes the query performance score predicted by our model with the parameter set θ for the given query.

Pairwise learning. Since the task of query performance prediction is often defined and evaluated as a ranking task [11, 44, 57] (ranking queries with respect to their performances), we can train our model using a pairwise setting. Therefore, each training instance consists of a random pair of queries from the training set T. To this end, we employ hinge loss (max-margin loss function) that has been widely used in the learning to rank literature for pairwise models [24]. Hinge loss is a linear loss function that penalizes examples violating the margin constraint. The hinge loss for a query pair q_i and q_j is defined as follows:

$$\mathcal{L}_k(q_i, q_j) = \max\{0, 1 - \text{sign}(\mathcal{Y}_{ik} - \mathcal{Y}_{jk}) (\widehat{P}_k(q_i; \mathcal{M}, C, \theta) - \widehat{P}_k(q_j; \mathcal{M}, C, \theta))\} \tag{9}$$

In the next subsection, we describe how each $\widehat{P}_k$ is computed.

4.2 Component Dropout

In training our model with multiple weak labels, we are faced with two major issues: (1) The predictions $\widehat{P}_1, \widehat{P}_2, \cdots, \widehat{P}_N$ should not be equal; otherwise, this would be equivalent to aggregating the weak labels and training the model using the aggregated labels. On the other hand, $\widehat{P}_1, \widehat{P}_2, \cdots, \widehat{P}_N$ should be produced by a single model that would be used at inference time. (2) As can be seen in Section 4.3, some of the employed weak supervision signals can be exactly computed using a sub-network of NeuralQPP. For instance, the retrieval scores analyzer component can compute NQC. Therefore, when we use NQC as a weak supervision signal, the network tries to predict the output only based on the retrieval scores analyzer component. This prevents the model from generalizing well. To overcome these two issues, we propose a component dropout approach that also regularizes our model.

Assume that we aim at training a NeuralQPP model with K components using N weak supervision signals. Let p^{ik}_{drop} denote the probability of dropping the effect of the i^{th} component for the k^{th} weak supervision signal ($1 \leq i \leq K$ and $1 \leq k \leq N$).

For each training instance, we construct a binary matrix B with the dimensionality of $K \times N$, whose elements are sampled from Bernoulli distributions as follows:

$$B_{ik} \sim \text{Bern}(1 - p^{ik}_{drop}) \tag{10}$$

Each element B_{ik} indicates whether the i^{th} component should be kept for the k^{th} weak supervision signal or not. We make sure that at least one component is kept, such that $\sum_{i=1}^{K} B_{ik} > 0$. Therefore, we compute each prediction $\widehat{P}_k$ as $\Gamma(\Lambda_k(\rho_1, \rho_2, \cdots, \rho_K))$, where Λ_k at training time is computed as:

$$\Lambda_k(\rho_1, \rho_2, \cdots, \rho_K) = \frac{1}{\sum_{i=1}^{K} B_{ik}} \sum_{i=1}^{K} B_{ik} \omega_i \rho_i \tag{11}$$

This results in different predictions for $\widehat{P}_1, \widehat{P}_2, \cdots, \widehat{P}_N$ at training time. At the inference time, no component must be dropped, so the matrix B is filled with 1s. In this case, Equation (11) is equivalent to Equation (2).

The proposed component dropout technique is similar to the field-level dropout approach, recently proposed by Zamani et al. [53] to prevent over-dependence on high-precision fields (e.g., clicked queries) in neural ranking models for semi-structured documents. The presented technique not also avoids overfitting on a weak supervision signal, but also allows us to use multiple weak signals as described earlier in this section.

4.3 Weak Supervision Signals

To train a generalized model, a natural decision would be to select weak labelers based on different intuitions, assumptions, and consumed information. This enables the neural model to observe complementary information in order to improve its generalization. Hence, we select a clarity-based approach, a score-based approach, and a combining approach (see Section 2.1 for more information about these categories) as the weak supervisors for our NeuralQPP model. The chosen weak labelers are described below:

Clarity. Clarity [11] is one of the early methods for query performance prediction that is based on the language modeling framework [35]. In more detail, this method estimates the query performance as follows:

$$clarity(q; C, \mathcal{M}) = \sum_{w \in V} p(w|\mathcal{R}_q) \log \frac{p(w|\mathcal{R}_q)}{p(w|\theta_C)} \tag{12}$$

where V denotes the vocabulary set, $\mathcal{R}_q$ represents the query language model estimated using relevance models [23], and θ_C represents the reference language model estimated using a maximum

[1] In our experiments, MAE resulted in more robust performance when compared to mean squared error (MSE).

likelihood estimation over the whole collection. Intuitively, this model measures the coherence of term distributions in the top retrieved documents with respect to the collection. The term distribution analyzer component (see Figure 1b) is expected to learn such a measurement. To generate this weak label, we set the number of retrieved documents to 200.

Normalized Query Commitment (NQC). NQC [44] measures the query performance by computing the normalized standard deviation of the retrieval scores assigned to the top retrieved documents, as follows:

$$NQC(q;C,\mathcal{M}) = \frac{\sqrt{\frac{1}{n}\sum_{D\in\pi^n_{\mathcal{M}}(q;C)}(\text{score}(q,D)-\hat{\mu})^2}}{\text{score}(q,C)} \quad (13)$$

where $\pi^n_{\mathcal{M}}(q;C)$ is the result list containing the top n retrieved documents in response to the query q. $\hat{\mu}$ denotes the mean retrieval scores in $\pi^n_{\mathcal{M}}(q;C)$. The intuition behind this model is that query drift can potentially be estimated by measuring the diversity of the retrieval scores. The retrieval scores analyzer component (see Figure 1a) also gives us such a measurement. To generate this weak label, the number of retrieved documents is again set to 200.

Utility Estimation Framework (UEF). UEF [42] is a theoretical framework by Shtok et al. based on statistical decision theory. UEF estimates the utility that each retrieved document provides w.r.t. the initiated query, as follows:

$$UEF(q;C,\mathcal{M}) \approx \text{sim}(\pi^n_{\mathcal{M}}(q;C), \pi^n_{\mathcal{M}}(\mathcal{R}_q;\pi^n_{\mathcal{M}}(q;C)))\ \Pr(\mathcal{R}_q|I_q) \quad (14)$$

where $\pi^n_{\mathcal{M}}(\mathcal{R}_q;\pi^n_{\mathcal{M}}(q;C))$ is the original result list re-ranked by the relevance model's estimation of the query language model ($\mathcal{R}_q$). The function 'sim' computes the similarity between two rank lists. We used Pearson's ρ coefficient as a ranking similarity measurement, as is the standard for QPP comparisons. To estimate the representativeness probability $\Pr(\mathcal{R}_q|I_q)$, we used Zhou and Croft's WIG approach [57] for the unigram language model[2]. It is computed as follows:

$$\Pr_{WIG}(\mathcal{R}_q|I_q) \propto \frac{1}{\sqrt{|q|}}\frac{1}{n}\sum_{D\in\pi^n_{\mathcal{M}}(q;C)}(\text{score}(q,D)-\text{score}(q,C)) \quad (15)$$

Note that the original UEF approach uses multiple samples to produce a relevance model, although, we used a single sampling to produce this weak label. To obtain this weak label, n is set to 100.

5 EXPERIMENTS

In this section, we study the effectiveness of the proposed method experimentally. We first introduce our datasets and then explain how our model is evaluated. We then describe our experimental setup in detail for further reproducibility. We finally discuss our empirical results.

5.1 Data

Collections. We evaluate our models using four TREC collections: The first two collections (AP and Robust) consist of thousands of news articles and are considered homogeneous collections. AP and Robust were previously used in the TREC 1-3 Ad-Hoc Tracks and the TREC 2004 Robust Track, respectively. The second two collections (GOV2 and ClueWeb) are large-scale web collections containing heterogeneous documents. GOV2 consists of the ".gov" domain web pages, crawled in 2004. ClueWeb (i.e., ClueWeb09-Category B) is a common web crawl collection that only contains English web pages. GOV2 and ClueWeb were previously used in TREC 2004-2006 Terabyte Track and TREC 2009-2012 Web Track, respectively. The statistics of these collections as well as the corresponding TREC topics are reported in Table 1. We use only the topic titles as queries.

We cleaned the ClueWeb collection by filtering out the spam documents. The spam filtering phase was done using the Waterloo spam scorer[3] [8] with the threshold of 60%. Stopwords were removed from all collections and no stemming was performed.

Training Queries. Similar to prior work on weak supervision for IR [16, 50], we computed all of the weak supervision signals (see Section 4.3) using several million unique queries obtained from the publicly available AOL query logs [32]. This dataset contains a sample of web search queries submitted to the AOL search engine within a three-month period from March 1, 2006 to May 31, 2006. We only used the query strings, and no session and click information was obtained from the query logs. We filtered out the navigational queries containing URL substrings, i.e., "http", "www.", ".com", ".net", ".org", ".edu". All non-alphanumeric characters were removed from the queries. As a sanity check, we made sure that no queries from the training set appear in our evaluation query sets. Applying all of these constraints leads to over 6 million unique queries as our training query set.

5.2 Evaluation

Following prior work on query performance prediction [11, 14, 40, 42, 44, 57], we evaluate our models by computing the correlation between the predicted performance and the actual average precision for the top 1000 documents retrieved per query (AP@1000). In our main experiment, we also report the correlation with the true NDCG values [20] for the top 20 documents. Note that NDCG@20 is a preferred evaluation metric for the ClueWeb collection due to the shallow pooling performed during relevance assessments [6, 25]. We predict the performance of the query likelihood model [35] with Dirichlet prior smoothing ($\mu = 1500$) [55] implemented in the Galago[4] search engine [10].

We use the standard measures from previous research to compute the correlation between predictions and actual performance. Pearson's ρ coefficient as a linear correlation metric and Kendall's τ coefficient as a ranking-based correlation metric are used. Statistically significant results are reported for two confidence intervals: 95% ($p\_value < 0.05$) and 99% ($p\_value < 0.01$).

Following prior work [40, 43, 44], to evaluate our models as well as the baselines, we first generate 30 equal-size random splits for each collection. In each split, the first fold is used for hyper-parameter optimization using grid search; the hyper-parameter setting that led to the highest Pearson's ρ correlation on predicting

[2]The original WIG approach is based on the term dependence model [28]. This bag-of-words variant is used as our third weak signal, and has been shown to be highly effective [5].

[3]http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

[4]http://www.lemurproject.org/galago.php

Table 1: Statistical properties of the four collections used.

ID	collection	queries (title only)	#docs	avg doc length	#qrels
AP	Associated Press 88-89	TREC 1-3 Ad-Hoc Track, topics 51-200	165k	287	15,838
Robust	TREC Disks 4 & 5 minus CR	TREC 2004 Robust Track, topics 301-450 & 601-700	528k	254	17,412
GOV2	2004 crawl of .gov domain	TREC 2004-2006 Terabyte Track, topics 701-850	25m	648	26,917
ClueWeb	ClueWeb 09 - Category B	TREC 2009-2012 Web Track, topics 1-200	50m	1506	18,771

the actual AP@1000 values was selected for evaluation on the second fold. This process was repeated for all 30 splits, and the average performance over the second folds are reported. This enables us to perform the paired t-test with Bonferroni correction to identify statistically significant differences between the performance of two QPP models ($p\_value < 0.05$).

5.3 Experimental Setup

We implemented and trained our models using TensorFlow[5]. The network parameters were optimized with the Adam optimizer [21] based on the back-propagation algorithm [41]. In our experiments, the learning rate was selected from $[1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3]$ and the batch size was set to 128. Either two or three hidden layers were used for each sub-networks of the NeuralQPP framework. The layer sizes were selected from $\{100, 300, 500\}$. We select the parameter vector α (see Equation (7)) from $\{0.2, 0.4, 0.6, 0.8\}$ and the dropout and the component dropout probabilities (see Equation (10)) from $\{0, 0.2, 0.4, 0.6, 0.8\}$. We initialized the embedding matrix $\mathcal{E}$ (see Equation (5)) by pre-trained GloVe [33] vectors trained on Wikipedia dump 2014 plus Gigawords 5.[6] The embedding dimension was set to 100.

5.4 Results and Discussions

In this section, we first evaluate our model against state-of-the-art unsupervised QPP approaches. We then analyze each component of the designed neural network. We further study the influence of incorporating multiple weak supervision signals in the NeuralQPP model. In our final experiments, we explore how NeuralQPP performs in terms of predicting the performance of various retrieval models.

Comparison with the Baselines. In the first set of experiments, we evaluate our models against popular and state-of-the-art query performance prediction baselines, including:

- Clarity [11]: See Section 4.3 for the details of this model.
- Query Feedback (QF): a high-performing QPP approach by Zhou and Croft [57] that measures the intersection of the result lists obtained by the original query and an estimated query from the top retrieved documents. Intuitively, this approach looks at the retrieval engine as a noisy channel and estimates the quality of the channel by measuring the amount of corruption in the result lists.
- Weighted Information Gain (WIG): a popular approach introduced by Zhou and Croft [57] that computes the information gain of the top retrieved documents compared to the collection. WIG predicts the query performance by analyzing the mean retrieval score and the collection's score.
- σ_k: a simple yet effective QPP model that computes the standard deviation of the retrieval scores for the top k retrieved documents. This model has been explored by Pérez-Iglesias and Araujo [34].
- $n(\sigma_{x\%})$: another approach based on the standard deviation, proposed by Cummins et al. [14], that uses a dynamic number of documents per query. This approach computes the standard deviation of the top retrieved documents whose retrieval scores are at least $x\%$ of the one obtained by the highest ranked document.
- Normalized Query Commitment (NQC) [44]: See Section 4.3 for the details of this model.
- Score Magnitude and Variance (SMV): a more recent QPP approach by Tao and Wu [46] that considers not only the "variance"[7] over the retrieval scores, but also the score magnitude.
- Robust Standard Deviation (RSD): a recent QPP method proposed by Roitman et al. [40] that computes multiple weighted standard deviations based on a bootstrapping approach. As suggested by the authors, we used WIG [57], as the sample weighting function.
- CombSum: a simple aggregation approach applied on top of the predictions generated by all the above baselines. For this model, we first normalize the scores generated by each model. This is a linear combining approach.
- Utility Estimation Framework (UEF) [42]: See Section 4.3 for the details of this model. The choice of representativeness probability in UEF is considered as a hyper-parameter and selected from {NQC, QF, WIG}.

As described in Section 5.2, all of the hyper-parameters of the baselines were optimized in the same way as the proposed models. In particular, the number of top retrieved documents is a common hyper-parameter in all of them. We selected this hyper-parameter from $\{5, 10, 15, 20, 25, 50, 100, 300, 500, 1000\}$.

The results for the above baselines and the proposed NeuralQPP model with two training settings (pointwise and pairwise) are reported in Table 2. Note that neither the baselines nor the proposed approaches require labeled training data. To have a fair comparison, we do not compare against supervised baselines, such as [36, 39]. The first observation from Table 2 is that there is no clear winner among the baselines. From the baseline results, predicting the query performance on the web collections is generally a much harder task when compared to the newswire collections. This is mostly due to the collection size, the variety of topics it covers, and the amount of noise in the collection. Although previous work mostly focused on predicting the performance of queries in terms of average precision for a deep ranking cut-off, we also provide the results for

[5] http://tensorflow.org/
[6] https://nlp.stanford.edu/projects/glove/

[7] SMV does not exactly compute the variance. Instead, its formulation is more similar to WIG [57].

Table 2: Performance of query performance prediction models on four collections, in terms of the Pearson's ρ and the Kendall's τ correlations. The results are reported for estimating the performance of each query in terms of two target metrics (AP@1000 and NDCG@20). The highest value in each column is marked in bold, and the superscripts † / ‡ denote statistically significant improvements compared to all baselines at 95% / 99% confidence intervals.

Method	Target Metric: AP@1000								Target Metric: NDCG@20							
	AP		Robust		GOV2		ClueWeb		AP		Robust		GOV2		ClueWeb	
	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ
Clarity	0.556	0.428	0.410	0.292	0.319	0.205	0.046	0.068	0.437	0.293	0.321	0.221	0.097	0.073	0.040	0.050
QF	0.585	0.438	0.418	0.274	0.494	0.324	0.273	0.130	0.442	0.296	0.379	0.263	0.322	0.208	0.205	0.084
WIG	0.547	0.391	0.444	0.294	0.462	0.321	0.238	0.202	0.427	0.287	0.335	0.207	0.308	0.225	0.255	0.175
σ_k	0.514	0.349	0.438	0.271	0.341	0.292	0.323	0.183	0.428	0.260	0.365	0.240	0.252	0.223	0.300	0.127
$n(\sigma_{x\%})$	0.524	0.289	0.380	0.218	0.342	0.255	0.188	0.139	0.428	0.210	0.273	0.158	0.232	0.196	0.185	0.082
NQC	0.540	0.369	0.445	0.283	0.424	0.321	0.308	0.139	0.464	0.290	0.390	0.255	0.257	0.203	0.270	0.102
SMV	0.505	0.349	0.401	0.274	0.357	0.279	0.326	0.156	0.438	0.266	0.371	0.256	0.335	**0.241**	0.282	0.121
RSD	0.594	0.406	0.455	0.352	0.444	0.276	0.193	0.096	0.459	0.315	0.394	0.286	0.339	0.203	0.199	0.095
CombSum	0.584	0.444	0.483	0.338	0.486	0.317	0.313	0.151	0.470	0.318	0.434	0.334	0.349	0.213	0.286	0.162
UEF	0.647	0.468	0.565	0.364	0.502	0.315	0.341	0.195	0.435	0.302	0.501	0.332	0.311	0.188	0.300	0.159
NeuralQPP (Pointwise)	0.613	0.432	0.582†	0.370	0.517	0.322	0.362‡	0.219	0.442	0.321	0.528†	**0.350**†	0.346	0.232	0.341‡	0.201‡
NeuralQPP (Pairwise)	**0.697**‡	**0.483**‡	**0.611**‡	**0.408**‡	**0.540**‡	**0.357**‡	**0.367**‡	**0.229**†	**0.492**‡	**0.336**†	**0.539**‡	0.343†	**0.371**†	0.239	**0.352**‡	**0.218**‡

an additional evaluation metric (NDCG@20) that computes the query performance for a shallow ranking cut-off. An interesting observation here is that estimating the query performance in terms of NDCG@20 is a harder task, since the predicted performance of various methods achieve a lower correlation with the actual NDCG@20 values in comparison with the AP@1000 values.

Our second observation from Table 2 is that the pairwise setting in the NeuralQPP model works much better than the pointwise setting. The reason might be related to the nature of the labels we use for training our models. In fact, weak supervision provides a set of noisy labels, and maximizing the likelihood of generating the labels by a neural model is not necessarily a proper choice; instead, optimizing a pairwise loss function gives more freedom to the model to obtain useful features to discriminate two queries. This enables the model to perform much better than the weak labels in almost all cases. A similar observation was made by Dehghani et al. [16] when training neural ranking models with weak supervision signals in the ad-hoc retrieval task.

Our third observation from the results reported in Table 2 is that NeuralQPP outperforms all the baselines, including the combining approaches, for most collections. The improvements achieved by the NeuralQPP model trained with a pairwise loss function are statistically significant in nearly all cases. This indicates the effectiveness of the proposed neural model and training for query performance prediction.

For the sake of space, we, hereafter, only focus on predicting AP@1000 for each query (which is also what previous work does [11, 22, 44, 57]). We also focus on the pairwise setting to train our model, which has superior performance.

Analysis of the NeuralQPP Components. As pointed out earlier, we propose three components to develop the NeuralQPP model (see Section 3). In this set of experiments, we evaluate the performance of each of these components, individually. We also evaluate the effectiveness of the proposed component dropout technique to regularize the model with multiple components. In this experiment, we train our model with the pairwise setting and with all weak labels.

Table 3 reports the results for this experiment. According to the results, the performance achieved by each of the individual components exhibits high variance. For instance, the term distribution analyzer component achieved the highest performance on the AP collection, however, the performance achieved by the retrieval scores analyzer component on the ClueWeb collection are far higher than those achieved by the other two components. This shows that various components can capture different aspects required for achieving improved performance on different collections. Our results also validate that our model successfully makes use of the information captured by multiple components – employing all components together outperforms all individual components. Furthermore, the results indicate that the component dropout technique is effective in all cases and leads to improved performance. All of the improvements obtained by NeuralQPP with all three components and with the component dropout technique are statistically significant when compared to each individual component.

Analysis of the Weak Supervision Signals. In the next set of experiments, we empirically study how employing multiple weak supervision signals (see Section 4.3) affects the NeuralQPP performance. To achieve this aim, we use our model with all three components trained by each of the weak supervision signals, individually. The results obtained by these models are reported in the first section of Table 4. In the second section, we present the results for two simple models that consider all weak signals (see Section 4.1 for more detail). The first model, All-MV, aggregates

Table 3: Performance of the NeuralQPP's individual components as well as the Component Dropout technique in case of existing multiple components. The Pearson's ρ and the Kendall's τ correlations are reported for the AP of the top 1000 documents per query. The highest value in each column is marked in bold, and the superscripts ‡ denotes statistically significant improvements compared to all individual components at a 99% confidence interval.

Component(s)	AP		Robust		GOV2		ClueWeb	
	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ
Retrieval score analyzer	0.536	0.388	0.442	0.289	0.351	0.280	0.346	0.188
Term distribution analyzer	0.541	0.447	0.419	0.319	0.308	0.212	0.056	0.073
Semantic Analyzer	0.471	0.353	0.485	0.307	0.378	0.210	0.090	0.084
All without Component Dropout	0.636‡	0.462	0.571‡	0.367‡	0.485‡	0.308‡	0.349	0.193
All with Component Dropout	**0.697**‡	**0.483**‡	**0.611**‡	**0.408**‡	**0.540**‡	**0.357**‡	**0.367**‡	**0.229**‡

Table 4: Performance of NeuralQPP trained with different weak labels, in terms of correlation with the actual AP@1000 values. The highest value in each column is marked in bold, and the superscript † / ‡ denote statistically significant improvements compared to all individual weak signals as well as both All-MV and All-Ind methods at 95% / 99% intervals.

Weak	AP		Robust		GOV2		ClueWeb	
Label	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ
Clarity	0.581	0.443	0.437	0.361	0.330	0.268	0.095	0.103
NQC	0.572	0.369	0.461	0.312	0.439	0.336	0.353	0.184
UEF	0.682	0.480	0.597	0.381	0.527	0.334	0.348	0.191
All-MV	0.694	0.477	0.520	0.351	0.454	0.316	0.357	0.187
All-Ind	0.591	0.454	0.447	0.362	0.384	0.316	0.116	0.128
All-CD	**0.697**	**0.483**	**0.611**†	**0.408**†	**0.540**†	**0.357**‡	**0.367**†	**0.229**‡

the outputs for all of the weak labelers. In fact, for each training pair, All-MV selects the label by majority voting over the output of all weak labelers. The second model, All-Ind, learns three separate NeuralQPP models, each by a single weak label, and then produces the final prediction by summing the output of these individually learned models. In the last section of the table, we report the results achieved by our model, All-CD (CD stands for component dropout).

We report the results of this experiment in Table 4. By looking at the results presented in both Tables 2 and 4, we can observe a clear correlation between the performance obtained by each method: Clarity, NQC, and UEF (see Table 2), and those achieved by NeuralQPP trained with each of these models as a weak signal (see Table 4), respectively. For example, NeuralQPP trained with the Clarity model as the weak signal performs well on the newswire collections, compared to the web collections. The Clarity method itself also behaves similarly. Table 4 also demonstrates that training with multiple weak signals leads to a higher generalization, and thus a more accurate performance predictor. These improvements are statistically significant on the Robust, GOV2, and ClueWeb collections.

Our results also demonstrate that the proposed approach for learning from multiple weak labelers is more effective than both All-MV and All-Ind. In particular, All-Ind has poor overall performance, because the models are learned individually and the scale of their outputs are not necessarily in the same scale.[8] Therefore, one of the models may bias the final prediction. As mentioned in Section 4.1 both All-MV and All-Ind suffer from information loss provided by multiple weak labels.

Table 5: Performance of the NeuralQPP model for predicting the average precision of the top 1000 documents for popular retrieval models.

Retr.	AP		Robust		GOV2		ClueWeb	
Model	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ	P-ρ	K-τ
QL	0.697	0.483	0.611	0.408	0.540	0.357	0.367	0.229
TF-IDF	0.671	0.480	0.619	0.412	0.562	0.386	0.355	0.210
BM25	0.718	0.503	0.624	0.412	0.483	0.322	0.310	0.197

Predicting the Performance of Various Retrieval Models. In the last set of experiments, we study the ability of NeuralQPP to predict the performance of various retrieval models. The results for evaluating the performance of three popular retrieval models: query likelihood (QL) [35], BM25, and TF-IDF. The results reported in Table 5 demonstrate that the NeuralQPP performs well in predicting the performance of all of these retrieval models.

6 CONCLUSIONS

In this paper, we proposed NeuralQPP, a general neural framework for the fundamental task of query performance prediction in ad-hoc retrieval. We implemented NeuralQPP using three components. The first two components analyze the retrieval score distribution and the term distribution in the result list, respectively. The third component, called the semantic analyzer, learns an abstract representation from the content of the top ranked documents in a semantic space. Due to the lack of training data in various settings, we proposed model training with weak supervision, an unsupervised learning approach that obtains training labels from existing unsupervised performance predictors. We explored how to train our model with multiple weak labels, which also led to the development of a component dropout technique to prevent overfitting on any of the weak supervision signals. We evaluated our models using four standard TREC collections, including two newswire and two

[8] Normalizing the scores for each split improves the performance for All-Ind, however, it performs worse than All-MV. The reason is that All-Ind components are trained separately.

large-scale web collections. The experiments demonstrated significant improvements over the baselines, in nearly every case. We also studied the contributions from each component in NeuralQPP, the effectiveness of employing multiple weak signals, and the positive effect of the component dropout technique on the performance prediction accuracy.

Acknowledgements. This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-1160894, in part by NSF grant #IIS-1419693, and in part by ARC grant DP170102231. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors. We thank Oren Kurland for insights on shallow evaluation limitations in QPPs.

REFERENCES

[1] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. 2011. Pseudo Test Collections for Learning Web Search Ranking Functions. In *Proc. SIGIR*. 1073–1082.
[2] J. A. Aslam and V. Pavlu. 2007. Query Hardness Estimation Using Jensen-Shannon Divergence Among Multiple Scoring Functions. In *Proc. ECIR*. 198–209.
[3] L. Azzopardi, M. de Rijke, and K. Balog. 2007. Building Simulated Queries for Known-item Topics: An Analysis Using Six European Languages. In *Proc. SIGIR*. 455–462.
[4] J. G. C. de Souza, H. Zamani, M. Negri, M. Turchi, and D. Falavigna. 2015. Multitask Learning for Adaptive Quality Estimation of Automatically Transcribed Utterances. In *Proc. NAACL*. 714–724.
[5] D. Carmel and E. Yom-Tov. 2010. *Estimating the Query Difficulty for Information Retrieval* (1st ed.). Morgan and Claypool Publishers.
[6] C. L. A. Clarke, N. Craswell, and E. M. Voorhees. 2012. Overview of the TREC 2012 Web Track. In *Proc. TREC*.
[7] D. Cohen, J. Foley, H. Zamani, J. Allan, and W. B. Croft. 2018. Universal Approximation Functions for Fast Learning to Rank. In *Proc. SIGIR*. (to appear).
[8] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. 2011. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Inf. Retr.* 14, 5 (Oct. 2011).
[9] W. B. Croft and D. J. Harper. 1979. Using Probabilistic Models of Document Retrieval Without Relevance Information. *J. Doc.* 35, 4 (1979), 285–295.
[10] W. B. Croft, D. Metzler, and T. Strohman. 2009. *Search Engines: Information Retrieval in Practice* (1st ed.). Addison-Wesley Publishing Company.
[11] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. 2002. Predicting Query Performance. In *Proc. SIGIR*. 299–306.
[12] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. 2006. Precision Prediction Based on Ranked List Coherence. *Inf. Retr.* 9, 6 (Dec. 2006), 723–755.
[13] J. S. Culpepper, C. L. A. Clarke, and J. Lin. 2016. Dynamic Cutoff Prediction in Multi-Stage Retrieval Systems. In *Proc. ADCS*. 17–24.
[14] R. Cummins, J. Jose, and C. O'Riordan. 2011. Improved Query Performance Prediction Using Standard Deviation. In *Proc. SIGIR*. 1089–1090.
[15] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps. 2017. Avoiding Your Teacher's Mistakes: Training Neural Networks with Controlled Weak Supervision. *CoRR* abs/1711.00313 (2017).
[16] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proc. SIGIR*. 65–74.
[17] F. Diaz. 2007. Performance Prediction Using Spatial Autocorrelation. In *Proc. SIGIR*. 583–590.
[18] C. Hauff, D. Hiemstra, and F. de Jong. 2008. A Survey of Pre-retrieval Query Performance Predictors. In *Proc. CIKM*. 1419–1420.
[19] J. He, M. Larson, and M. de Rijke. 2008. Using Coherence-based Measures to Predict Query Difficulty. In *Proc. ECIR*. 689–694.
[20] K. Järvelin and J. Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446.
[21] D. P. Kingma and J. Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*.
[22] O. Kurland, A. Shtok, D. Carmel, and S. Hummel. 2011. A Unified Framework for Post-retrieval Query-performance Prediction. In *Proc. ICTIR*. 15–26.
[23] V. Lavrenko and W. B. Croft. 2001. Relevance Based Language Models. In *Proc. SIGIR*. 120–127.
[24] H. Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers.
[25] X. Lu, A. Moffat, and J. S. Culpepper. 2016. The effect of pooling and evaluation depth on IR metrics. *Inf. Retr.* 19, 4 (2016), 416–445.
[26] C. Luo, Y. Zheng, J. Mao, Y. Liu, M. Zhang, and S. Ma. 2017. Training Deep Ranking Model with Weak Relevance Labels. In *Proc. ADC*. 205–216.
[27] J. Mackenzie, J. S. Culpepper, R. Blanco, M. Crane, C. L. A. Clarke, and J. Lin. 2018. Query Driven Algorithm Selection in Early Stage Retrieval.. In *Proc. WSDM*. 396–404.
[28] D. Metzler and W. B. Croft. 2005. A Markov Random Field Model for Term Dependencies. In *Proc. SIGIR*. 472–479.
[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. NIPS*. 3111–3119.
[30] J. Mothe and L. Tanguy. 2004. Linguistic Features to Predict Query Difficulty. In *Proc. SIGIR Workshop on predicting Query Difficulty - Methods and Applications*.
[31] M. Negri, M. Turchi, J. G. C. de Souza, and D. Falavigna. 2014. Quality Estimation for Automatic Speech Recognition. In *Proc. COLING*. 1813–1823.
[32] G. Pass, A. Chowdhury, and C. Torgeson. 2006. A Picture of Search. In *Proc. InfoScale*.
[33] J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proc. EMNLP*. 1532–1543.
[34] J. Pérez-Iglesias and L. Araujo. 2010. Standard Deviation As a Query Hardness Estimator. In *Proc. SPIRE*. 207–212.
[35] J. M. Ponte and W. B. Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proc. SIGIR*. 275–281.
[36] F. Raiber and O. Kurland. 2014. Query-performance Prediction: Setting the Expectations Straight. In *Proc. SIGIR*. 13–22.
[37] J. J. Rocchio. 1971. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall.
[38] H. Roitman. 2017. An Enhanced Approach to Query Performance Prediction Using Reference Lists. In *Proc. SIGIR*. 869–872.
[39] H. Roitman, S. Erera, O. Sar-Shalom, and B. Weiner. 2017. Enhanced Mean Retrieval Score Estimation for Query Performance Prediction. In *Proc. ICTIR*. 35–42.
[40] H. Roitman, S. Erera, and B. Weiner. 2017. Robust Standard Deviation Estimation for Query Performance Prediction. In *Proc. ICTIR*. 245–248.
[41] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning Representations by Back-propagating Errors. *Nature* 323 (1986), 533–536.
[42] A. Shtok, O. Kurland, and D. Carmel. 2010. Using Statistical Decision Theory and Relevance Models for Query-performance Prediction. In *Proc. SIGIR*. 259–266.
[43] A. Shtok, O. Kurland, and D. Carmel. 2016. Query Performance Prediction Using Reference Lists. *ACM Trans. Inf. Syst.* 34, 4 (June 2016), 19:1–19:34.
[44] A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits. 2012. Predicting Query Performance by Query-Drift Estimation. *ACM Trans. Inf. Syst.* 30, 2 (May 2012).
[45] L. Specia, M. Turchi, N. Cancedda, M. Dymetman, and N. Cristianini. 2009. Estimating the Sentence-Level Quality of Machine Translation Systems. In *Proc. EAMT*. 28–37.
[46] Y. Tao and S. Wu. 2014. Query Performance Prediction By Considering Score Magnitude and Variance Together. In *Proc. CIKM*. 1891–1894.
[47] P. Thomas, F. Scholer, P. Bailey, and A. Moffat. 2017. Tasks, Queries, and Rankers in Pre-Retrieval Performance Prediction. In *Proc. ADCS*. 11:1–11:4.
[48] C. J. van Rijsbergen. 1979. *Information Retrieval* (2nd ed.). Butterworth-Heinemann, Newton, MA, USA.
[49] V. Vinay, I. J. Cox, N. Milic-Frayling, and K. Wood. 2006. On Ranking the Effectiveness of Searches. In *Proc. SIGIR*. 398–404.
[50] H. Zamani and W. B. Croft. 2017. Relevance-based Word Embedding. In *Proc. SIGIR*. 505–514.
[51] H. Zamani, J. Dadashkarimi, A. Shakery, and W. B. Croft. 2016. Pseudo-Relevance Feedback Based on Matrix Factorization. In *Proc. CIKM*. 1483–1492.
[52] H. Zamani, M. Dehghani, F. Diaz, H. Li, and N. Craswell. 2018. SIGIR 2018 Workshop on Learning from Limited or Noisy Data for Information Retrieval. In *Proc. SIGIR*. (to appear).
[53] H. Zamani, B. Mitra, X. Song, N. Craswell, and S. Tiwary. 2018. Neural Ranking Models with Multiple Document Fields. In *Proc. WSDM*. 700–708.
[54] C. Zhai and J. Lafferty. 2001. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *Proc. CIKM*. 403–410.
[55] C. Zhai and J. Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.* 22, 2 (April 2004), 179–214.
[56] Y. Zhou and W. B. Croft. 2006. Ranking Robustness: A Novel Framework to Predict Query Performance. In *Proc. CIKM*. 567–574.
[57] Y. Zhou and W. B. Croft. 2007. Query Performance Prediction in Web Search Environments. In *Proc. SIGIR*. 543–550.