Query-Performance Prediction: Setting the Expectations Straight

Fiana Raiber
fiana@tx.technion.ac.il

Oren Kurland
kurland@ie.technion.ac.il

Faculty of Industrial Engineering and Management, Technion — Israel Institute of Technology

ABSTRACT

The query-performance prediction task has been described as estimating retrieval effectiveness in the absence of relevance judgments. The expectations throughout the years were that improved prediction techniques would translate to improved retrieval approaches. However, this has not yet happened. Herein we provide an in-depth analysis of why this is the case. To this end, we formalize the prediction task in the most general probabilistic terms. Using this formalism we draw novel connections between tasks — and methods used to address these tasks — in federated search, fusion-based retrieval, and query-performance prediction. Furthermore, using formal arguments we show that the ability to estimate the probability of effective retrieval with no relevance judgments (i.e., to predict performance) implies knowledge of how to perform effective retrieval. We also explain why the expectation that using previously proposed query-performance predictors would help to improve retrieval effectiveness was not realized. This is due to a misalignment with the actual goal for which these predictors were devised: ranking queries based on the presumed effectiveness of using them for retrieval over a corpus with a specific retrieval method. Focusing on this specific prediction task, namely query ranking by presumed effectiveness, we present a novel learning-to-rank-based approach that uses Markov Random Fields. The resultant prediction quality substantially transcends that of state-of-the-art predictors.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval models

Keywords: query-performance prediction, learning-to-rank

1. INTRODUCTION

The query-performance prediction task has attracted much research attention [11]. The goal of the task has been stated as estimating the effectiveness of retrieval performed in response to a query with no relevance judgments [11]. An important claim put forth for motivating the development of query-performance predictors was that using them would help to improve retrieval effectiveness.

However, the expectation that improved prediction methods would translate to improved retrieval approaches has not been realized. One of our goals here is to explore why this is the case. To this end, we formalize the prediction challenge in the most general probabilistic terms. Using this formalism we draw novel connections between tasks and methods used to address them in federated search [10], fusion-based retrieval [14] and query-performance prediction [11].

We formally show that the ability to successfully estimate the probability of effective retrieval with no relevance judgments (i.e., to predict performance) directly implies knowledge of how to perform effective retrieval. Thus, at the conceptual level, it should come as no surprise that improved prediction methods do not lead to improved retrieval methods. On the practical level, we make the observation that previously proposed query-performance prediction methods — specifically, those analyzing the retrieved list — were designed as approaches to ranking queries based on presumed retrieval effectiveness rather than as direct estimates for the probability of effective retrieval. Nevertheless, we argue for why even in their capacity as query ranking techniques the most effective among the proposed predictors were not, and cannot, be used to improve retrieval effectiveness. The core issue is that these predictors are by design effective only in ranking queries for specific retrieval methods.

Given that previously proposed query-performance predictors are essentially query ranking methods, we present a learning-to-rank approach based on Markov Random Fields for query-performance prediction. The resultant prediction quality substantially transcends that of state-of-the-art predictors. We then use our approach in a fusion-based retrieval setting to further support our claim that the expectation that using query-performance predictors would help to improve retrieval effectiveness is somewhat unrealistic. These predictors should be examined and evaluated with respect to the task they were designed for: estimating for which queries a retrieval performed over a given corpus with a given retrieval method would result in better performance.

2. RELATED WORK

There are three main lines of work on query-performance prediction. The first is devising pre-retrieval prediction methods that are based only on the query and corpus-based statistics [15, 25, 42, 37, 23, 53, 41]. The second is that on post-retrieval prediction approaches that analyze also the retrieved list [11]. Specifically, some methods measure the

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

http://dx.doi.org/10.1145/2600428.2609581.

clarity of the list with respect to the corpus [15, 16, 2, 24]. Others quantify different notions of the robustness of the list [52, 51, 54, 4, 40]. There are post-retrieval predictors that rely on analysis of the retrieval scores distribution in the list [50, 9, 19, 55, 6, 38, 17, 18, 46].

The third line of work is on prediction frameworks. One framework is based on measuring distances between query representations, relevant documents, and the corpus [12]. In another framework the prediction challenge was casted as a utility estimation task [45]. A probabilistic framework was proposed to explain the common grounds of various (pre- and post- retrieval) prediction methods [29]. The basic probability used in this work to derive the framework emerges as a specific case of the formalism we present here. In a similar vein, various post-retrieval predictors were presented as instantiations of a simple prediction principle [28]; namely, comparing the retrieved list with a pseudo (in)effective list.

In contrast to all this previous work, we formally show that the query-performance prediction task which has been addressed is a specific instance of a much more general prediction challenge. Accordingly, we establish novel connections between various prediction tasks. We also study the reasons to why improved query-performance prediction has not translated to improved novel retrieval methods.

It was argued using a simulation [21] that the state-of-the-art prediction quality should much improve to allow for effective selective query expansion [2, 16]. We provide here reasons to the potential difficulty in substantially improving prediction quality by connecting the ability to predict performance with the knowledge to perform effective retrieval.

Our learning-to-rank framework for query-performance prediction integrates previously proposed pre- and post- retrieval predictors. Pre-retrieval predictors were integrated using linear regression [22]. In contrast to our work, post-retrieval predictors which (as we show) are more effective than pre-retrieval predictors were not integrated. More generally, the prediction task was not treated as a ranking problem. Post- and pre- retrieval predictors were also integrated in a probabilistic framework [29]. In contrast to our learning-to-rank framework, only two post-retrieval predictors and one pre-retrieval predictor could be integrated; and, the integration was non parametric. Hence, the relative prediction quality of the predictors, as determined using a train set of queries, could not be utilized. Linear interpolation of only two post-retrieval predictors was also proposed [51, 55, 45, 28], but without applying a learning-to-rank approach.

3. THE PREDICTION TASK

The query performance prediction task was stated as "estimating the effectiveness of a search performed in response to a query in the absence of relevance judgements" [15, 11]. To formalize the task in the most general terms, we use the following notation. Q, C and M are random variables that take as values queries, document corpora, and retrieval methods, respectively. A retrieval method gets as input a query and a corpus of documents and produces a ranking over the corpus. The random variable R is assigned with a value 1 if the retrieval was effective and 0 otherwise. Accordingly, predicting retrieval effectiveness amounts to estimating

$$p(R = 1|Q, C, M). \quad (1)$$

Different assignments to the random variables in Equation 1 result in different prediction tasks that we discuss below. These tasks were not necessarily referred to in past work as prediction, nor derived from the same basic formalism. Table 1 summarizes these tasks.

3.1 Prediction over corpora

In the federated retrieval setting [10, 44], a set of corpora, $\{c_i\}$, can be used for retrieval. The resource selection problem is to select a subset of these corpora to perform the retrieval on [10, 44]. This task can be performed by addressing a prediction challenge *over corpora*; specifically, estimating for each corpus c the probability that using a given query q in *some* retrieval method upon this corpus will result in effective retrieval. Formally, we fix $Q = q$, and for each c ($\in \{c_i\}$) estimate $p(R = 1|Q = q, C = c, M)$, for any assignment of M. We re-write this probability as

$$p_q(R = 1|C = c), \quad (2)$$

to emphasize the fact that the query is fixed and that the estimation is performed across corpora and independently of a retrieval method.

In practice, the probability in Equation 2 is not estimated directly. Rather, the corpora are ranked in response to the query using some form of corpora representation [10, 44].

3.2 Prediction over retrieved lists

The fusion task is merging lists that were retrieved from the same corpus in response to a query [14]. The lists differ due to the retrieval methods employed to produce them; e.g., the document representation, the query representation, and the ranking function can be varied [14].

A long standing challenge in work on fusion is estimating the effectiveness of the lists to be fused in lack of relevance judgements [3]. Such estimates could be used, for example, to weigh the lists when linearly fusing them [3, 31, 43]; or, to select a single list as the final result list [2, 16, 34, 5].

Thus, estimating list effectiveness in the fusion setting is a prediction task performed *over retrieval methods*, and consequently, *over retrieved lists*. That is, the choice of a retrieval method for a given pair of a query and a corpus entails a ranking. Formally, we fix $Q = q$ and $C = c$ as retrieval is performed over a single corpus. Then, for each retrieval method m we follow Equation 1 and estimate $p(R = 1|Q = q, C = c, M = m)$. Doing that amounts to estimating $p(R = 1|Q = q, C = c, L = l)$ for each retrieved document list l, where L is a random variable that takes as values such lists. We write this probability as

$$p_{q;c}(R = 1|L = l), \quad (3)$$

to emphasize the fact that both the query and the corpus are fixed, and the estimation is across retrieved lists.

In previous work on fusion, the probability in Equation 3 was not directly estimated. Rather, the estimation task was often replaced with a "ranking retrieved lists with respect to a query" task for a fixed query and corpus. List ranking was performed based on past performance of the retrieval methods (e.g., [3, 31]), features assumed to be correlated with list effectiveness [34, 5, 43], or the similarity between the list and the "average list" among those available [47, 19].

3.3 Prediction over queries

Heretofore, we used the term "prediction" to refer to the tasks described above as they are manifestations of the challenge of estimating Equation 1 with no relevance judgments.

prediction task	target probability	fixed	varied	no effect
fundamental	$p(R=1\|Q,C,M)$	none	none	none
over corpora	$p_q(R=1\|C=c)$	Q	C	L
over lists	$p_{q;c}(R=1\|L=l)$	Q,C	L	none
(pre-) over queries	$p_c(R=1\|Q=q)$	C	Q	L
(post-) over queries	$p_c(R=1\|Q=q,L=l)$	C	Q,L	none

Table 1: Summary of the prediction tasks. "target probability": the probability to be estimated using no relevance judgments; "fixed", "varied", "no effect": the random variables whose values are (i) fixed, (ii) varied (and consequently, over which prediction is performed), and (iii) has no effect on the estimation, respectively. A triplet of a query (Q), corpus (C) and retrieval method (M) entails a retrieved list (L). "pre-" and "post-" stand for pre-retrieval and post-retrieval, respectively.

However, these tasks were not referred to as "prediction" in past literature. In fact, almost all methods that were proposed in past work on query-performance prediction address (either explicitly or implicitly) the challenge of *prediction over queries*. These methods can be categorized into the two classes that we discuss next [11].

Pre-retrieval prediction. The first class of prediction over queries approaches is that of pre-retrieval prediction. Given a query and a fixed corpus of documents, retrieval effectiveness is estimated before retrieval is employed and independently of a retrieval method [15, 25, 42, 37, 24, 53, 41]. Then, the estimates are compared across queries to predict which queries will result in more effective retrieval. Formally, C is fixed to c, and following Equation 1 the task becomes estimating $p(R=1|Q=q,C=c,M)$. To emphasize the fact that the corpus is fixed, and that the estimation is across queries and independent of a retrieval method (and therefore of a retrieved list), we write the probability as:

$$p_c(R=1|Q=q). \quad (4)$$

As was the case for the prediction tasks described in Sections 3.1 and 3.2, Equation 4 was not directly estimated in past work [15, 25, 42, 37, 24, 53, 41].[1] Rather, methods for ranking queries with respect to a corpus have been devised.

We note the interesting connection between Equations 2 and 4. Both are specific instantiations of Equation 1. In the prediction over corpora case (Equation 2), the query is fixed and the corpora are varied. In the pre-retrieval prediction over queries case (Equation 4), the corpus is fixed and the queries are varied. In both cases, the estimation (prediction) is for *any* retrieval method. Perhaps not surprisingly, then, effective methods for ranking corpora and queries (in a pre-retrieval mode) bear much resemblance. For example, the CORI resource selection approach for federated search [10] ranks corpora with respect to the query using a tf.idf-based similarity measure. The SCQ pre-retrieval over queries predictor scores queries with respect to a corpus also using a tf.idf-based similarity measure [53]. This connection between the formal tasks of resource selection and pre-retrieval prediction over queries, and methods used to address them, is novel to this study.

[1]Pre-retrieval predictors were used in a linear regression to predict average precision [22], but the probability of effective retrieval was not directly estimated.

Post-retrieval prediction. Methods referred to as post-retrieval query-performance predictors estimate retrieval effectiveness based on the query, the corpus, and the retrieved list [15, 2, 16, 50, 9, 52, 12, 51, 54, 4, 19, 55, 24, 6, 45, 17, 18, 46]. The goal is to compare estimates across queries as is the case for pre-retrieval prediction that was discussed above. We can formalize the task as follows. We fix a corpus, $C=c$. Then, following Equation 1 we estimate for each pair of a query (q) and a retrieval method (m) $p(R=1|Q=q,C=c,M=m)$. Given a corpus, a pair of a query and a retrieval method entails a retrieved list, l. Furthermore, most work on post-retrieval prediction has focused on analyzing the retrieved list, rather than the retrieval method itself.[2] Thus, the probability of interest is:

$$p_c(R=1|Q=q,L=l). \quad (5)$$

Equation 5 served as the basis for deriving a probabilistic framework that was used to explain the commonalities between prediction over queries methods [29]. Virtually all previously proposed post-retrieval predictors do not directly estimate Equation 5. Instead, scoring methods are devised for pairs of a query and a retrieved list. The scores are used to rank the queries by presumed retrieval effectiveness.

We also note the fundamental difference between Equations 5 and 3. In post-retrieval prediction over queries (Equation 5) estimation is performed across pairs of query and retrieved list. In the fusion setting (Equation 3), the query is fixed and estimation is performed across retrieved lists. We come back to this point below.

3.4 Improved prediction ⇒ improved retrieval?

Work on query-performance prediction — i.e., prediction over queries — based the motivation for devising predictors on the claim that improved prediction would help to improve retrieval effectiveness [11, Chapter 1]. However, there was very little empirical support to this claim.[3] We now explore the gap between the claim and the lack of its realization.

Consider Equation 1 that was used to derive the prediction tasks. Suppose that we fix the query, $Q=q$, and that the corpus contains a single document d which is the only one that can be retrieved by any retrieval method. Accordingly, the prediction task becomes estimating $p(R=1|Q=q,D=d)$; D is a random variable that takes documents as values. That is, the prediction task amounts to the core estimation challenge in probabilistic retrieval: *what is the probability that this document is relevant to this query?* [49]. Thus, we arrive to the following argument:

The ability to successfully estimate the probability of effective retrieval with no relevance judgments (i.e., predict performance) directly implies knowledge of how to perform effective retrieval.

[2]There is work on predicting retrieval effectiveness based on statistics of features used by the retrieval method [5, 6].

[3]Small performance improvements were attained when using (i) post-retrieval predictors in federated search [52] and (ii) pre-retrieval predictors in a learning-to-rank framework [35]. Post-retrieval predictors were also integrated with many (ranker specific) features to select a ranking function [5] (with somewhat inconclusive findings about the resultant effectiveness) and for fusion [43] (with small relative contribution to overall performance). Selective query expansion using predictors was not shown to be of much merit [2, 16].

A potential criticism about the argument just posed is that prediction methods are presumably more successful in estimating retrieval effectiveness for a retrieved document set than for a single document, because they exploit properties of such sets. To address this criticism, we can set the corpus in Equation 1 to contain only a cluster of similar documents. Work on ranking document clusters in response to a query [33, 39] showed that effective estimates for cluster relevance translate to improved document ranking. Thus, we get further support to the argument that the ability to predict the effectiveness of a retrieved document set implies knowledge of performing effective document retrieval.[4]

Another potential attack on the argument is that the inability to accurately estimate the probability of effective retrieval does not necessarily imply inability to effectively rank the items of concern. Indeed, in Sections 3.1, 3.2 and 3.3 we made the observations that in most cases the probability for effective retrieval was not estimated directly, but rather ranking approaches have been employed.

The situation just described is reminiscent of that in ad hoc retrieval. In probabilistic retrieval methods [49], the probability of document relevance is rarely directly estimated. Instead, quantities which are presumably rank equivalent to this probability are used to rank documents.

Hence, we re-visit the above mentioned motivation for devising prediction over queries methods from a ranking point of view. A recurring example for using these predictors to improve retrieval effectiveness — which was not backed up empirically — was switching a retrieval method if retrieval was predicted to fail [11]. Presumably, there is a task mismatch: post-retrieval over queries predictors rank pairs of a query and a retrieved list, while the suggested task is ranking retrieved lists for a given query (prediction over retrieved lists).[5] However, if we fix the query in post-retrieval over queries prediction methods we should presumably get retrieved-lists ranking methods. More generally, fixing the query in Equation 5 results in the post-retrieval over queries prediction task becoming the prediction over retrieved lists task (Equation 3).

Yet, as it turns out, the most effective previously proposed post-retrieval over queries predictors are (either explicitly or implicitly) coupled with, and therefore effective for, a specific retrieval method or family of retrieval methods. We explain why this is the case in Appendix A. Thus, it should not be expected that these prediction methods could help to select a retrieved list for a given query. In other words, these predictors are essentially query ranking functions which exploit information induced from the retrieved list and which are based on a specific retrieval approach. Indeed, the evaluation of the prediction quality of these methods in past work was based on fixing the corpus and the retrieval method — often, to a standard document ranking approach that uses document-query surface-level similarities — and measuring the effectiveness of the query ranking [11].

We draw the following conclusions. The ability to directly estimate, with no relevance judgments, the probability for effective retrieval implies the knowledge of how to perform effective retrieval. Thus, at this conceptual level, it should not be expected that improved prediction would translate to *novel* improved retrieval methods. On the practical level, the prediction tasks discussed above have been addressed using ranking functions rather than by directly estimating the probability for effective retrieval. Specifically, the most effective post-retrieval over queries predictors are query ranking methods that are committed to specific retrieval approaches. Hence, it should not be expected that using these predictors would help in selecting a retrieved list for a query so as to improve retrieval performance.

3.5 Learning to rank queries using Markov Random Fields

With the realization that previously proposed prediction over queries methods (a.k.a. query-performance predictors [11]) — both pre-retrieval and post-retrieval — are essentially query ranking methods, we now turn to tackle the prediction over queries task using a learning-to-rank approach.

Inspired by Equation 1, and given that the choice of a query and a retrieval method entails a retrieved list for a given corpus, we set as a goal to estimate

$$p(R = 1, Q, C, L). \tag{6}$$

This is the joint probability for a relevance event and a triplet of a query (Q), a corpus (C), and a document list (L) retrieved for the query from the corpus.

We estimate Equation 6 using Markov Random Fields (MRFs). An MRF is defined over a graph G with nodes representing random variables and edges representing dependencies between the variables. The nodes of the graph here are the query (Q), the corpus (C), and the retrieved list (L). As is standard practice in using MRFs to rank documents [36], we omit the relevance random variable (R) from the graph to simplify the formulation. This has no effect on the resultant query ranking. Accordingly, we re-write Equation 6 as $p_{R=1}(Q, C, L)$. This is the joint probability over G's nodes which can be written as:

$$p_{R=1}(Q, C, L) = \frac{1}{Z} \prod_{x \in X(G)} \psi_x(x); \tag{7}$$

$X(G)$ are the cliques in G; $\psi_x(x)$ is a potential (positive) function defined over the clique x; $Z = \sum_{Q,C,L} \prod_{x \in X(G)} \psi_x(x)$ is the normalization factor. Computing this factor is a very hard task. However, since our goal is to *rank* queries, and the normalization factor does not affect this ranking, we do not compute it. This is yet another example for the transition mentioned in Section 3.4 from devising direct estimates of probabilities to applying rank-equivalence manipulations that yield ranking functions.

We use the standard instantiation of potential functions [36]: $\psi_x(x) \stackrel{def}{=} \exp(\lambda_x f_x(x))$, where $f_x(x)$ is a feature function defined over the clique x and weighted by the parameter λ_x. Using the feature functions in Equation 7, omitting the normalization factor, and applying a log transformation, we arrive to a linear ranking function for queries:

$$Score(Q; C, L) \stackrel{def}{=} \sum_{x \in X(G)} \lambda_x f_x(x). \tag{8}$$

Most linear learning-to-rank methods are based on a linear combination of feature functions as in Equation 8 [32]. Our motivation to address the query ranking challenge using MRFs is the correspondence between the graph structure

[4]The connection between query-performance prediction and cluster ranking was also stated elsewhere [27].

[5]Pre-retrieval query-performance predictors are independent of the retrieved list and therefore cannot be used *alone* to select a retrieved list.

and the types of information used by previously proposed query-performance prediction approaches that serve for feature functions in Section 3.5.1.

Each feature function $f_x(x)$ that we use in Equation 8 is defined as $\log(h_x(x)+\epsilon)$, where h is one of the feature functions defined below; $\epsilon = 10^{-10}$ is a smoothing factor. This results in a conjunction style integration of feature functions similarly to work on using MRFs for document [36] and cluster [39] ranking. In Section 4.1.1 we use SVM^{rank} [26] to learn the values of the λ_x's in Equation 8.

3.5.1 Cliques and feature functions

We consider four different cliques in the graph G. These are depicted in Figure 1. In what follows, we describe the cliques and the feature functions associated with them. All feature functions are previously proposed query-performance predictors. Using these enables us to (i) demonstrate their correspondence with cliques; and, (ii) study whether a learning-to-rank method can effectively integrate them.

The x_{QC} clique. This clique is composed only of the query and the corpus. Thus, it corresponds to pre-retrieval query-performance predictors that do not analyze the retrieved list. We use three different measures for a query term that utilize corpus-based information. Then, we apply three functions: Sum, Average and Max, to aggregate the measures across the query terms. The first measure is the tf.idf-based similarity between a term and the corpus computed using the **SCQ** predictor [53]. The second measure is the variance of the tf.idf values of a term over the documents in the corpus in which it appears [53], referred to as **VAR**. The third measure is the inverse document frequency, **IDF**, of the term [16]. Using the SCQ-based measure amounts to assuming that query terms similar to the corpus indicate effective retrieval. The VAR and IDF measures are estimates of the discriminative power of the query terms. All together, we have 9 feature functions of the form "AB" for the x_{QC} clique, where $A \in$ {Sum,Avg,Max} and $B \in$ {SCQ,VAR,IDF}.

The x_L clique. This clique contains only the retrieved list. Feature functions associated with the clique reflect query-independent properties of the list that potentially attest to retrieval effectiveness.

The first feature function measures the **cohesion** of the retrieved list [12, 51]. The premise is that a cohesive list is more likely to be effective than a diverse one. To measure cohesion, we compute for each document d in L its similarity with all documents in L, and average the resultant values over all the documents in L. The inter-document similarity measure is described in Section 4.1.1.

The next feature functions are adopted from recent work on estimating document-list quality [40]. Three measures are defined for each document in the list. These potentially attest to the breadth of the document content. Content breadth can be thought of as a prior for document relevance [8]. The measures are then averaged over the documents in the retrieved list to yield the feature functions.

High entropy of the term distribution in the document potentially indicates content breadth [8]. The **entropy** of document d is defined as $-\sum_{w \in d} p(w|d) \log p(w|d)$; w is a term and $p(w|d)$ is the probability assigned to w by an unsmoothed unigram language model induced from d.

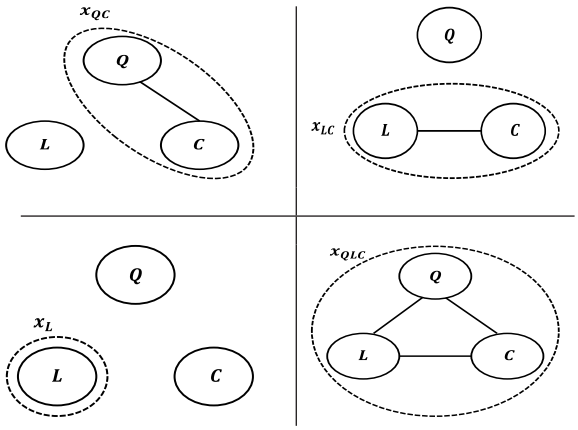

Figure 1: The four cliques considered for graph G. G is composed of a query node (Q), corpus node (C) and a retrieved list node (L).

Two additional feature functions are: (i) the ratio between the number of stopwords and non-stopwords in the document, **sw1**; and (ii) the fraction of stopwords in a stopword list (INQUERY's in our case [1]) that appear in the document, henceforth **sw2**. These functions were shown to be highly effective document quality measures, especially for noisy Web collections [8]. The reason is that the presence of stopwords presumably attests to the use of "rich" language and therefore to content breadth [8].

The x_{LC} clique. This clique includes the retrieved list and the corpus but not the query. Thus, it corresponds to query-independent properties of the retrieved list which are quantified using corpus-based information. One such property is **Clarity** [15]. This is the KL divergence between a relevance language model [30] induced from the list and that induced from the corpus. The premise is that a retrieved list with a language model different from that of the corpus is focused, and hence reflects effective retrieval. We also use **ImpClarity** which is a variant of Clarity proposed for Web corpora [24]: only the terms that appear in less than t% of the documents in the corpus are used to induce a relevance language model from the list; t is a free parameter.

The x_{QLC} clique. This clique contains all the nodes in the graph. The first two feature functions that we consider are the highly effective post-retrieval query-performance predictors **WIG** [55] and **NQC** [46]. WIG is the difference between the mean retrieval score in the list and that of the corpus which represents a pseudo non-relevant document.[6] The premise is that high retrieval scores with respect to that of the corpus attest to effective retrieval. NQC is defined as the standard deviation of retrieval scores in the list. High deviation was argued to correlate with potentially reduced query drift, and thus with improved effectiveness [46].[7]

The highly effective UEF prediction framework [45] is based on re-ranking the retrieved list L using a relevance

[6]The difference is normalized by the query length to ensure inter-query compatibility of prediction values.

[7]The standard deviation is normalized with respect to the corpus retrieval score to ensure inter-query compatibility of prediction values.

language model induced from L. We use the exponent of the Pearson correlation between the scores in L and those produced by the re-ranking as a basic prediction measure. The measure is scaled by the value assigned by some basic predictor — in our case, Clarity, ImpClarity, WIG or NQC— to produce the final prediction value. We get 4 feature functions: **UEF(Clarity)**, **UEF(ImpClarity)**, **UEF(WIG)** and **UEF(NQC)**. The core idea, as mentioned in Appendix A, is that the relevance-model-based re-ranking is more effective than the original ranking and can therefore serve as a reference comparison. The basic predictors serve to estimate the presumed quality of the relevance model.

4. EVALUATION

We start in Section 4.1 by evaluating the effectiveness of our learning-to-rank method, henceforth **LTRoq**, for the prediction over queries task. Then, in Section 4.2 we address the claim made in Section 3.4 that previously-proposed post-retrieval over queries predictors are not effective for prediction over retrieved lists.

4.1 Prediction over queries

4.1.1 Experimental setup

The TREC datasets used for the prediction over queries experiments are specified in Table 2. WSJ, AP and ROBUST are small collections, mostly composed of news articles. WT10G is a small Web collection, and GOV2 is a crawl of the .gov domain. For the ClueWeb09 collection we used both the Category B subset, CW09B, and the English part of Category A, CW09A. To study the effect of spam documents on prediction quality, we created an additional experimental setting for each category of ClueWeb09 [13]. Specifically, documents assigned with a score below 50 (70) by Waterloo's spam classifier [13] were filtered out from the initial ranking created over the documents in CW09B (CW09A); the residual corpus ranking was used for experiments. Accordingly, we get the additional CW09BF setting for Category B and the CW09AF setting for Category A.

corpus	# of docs	data	queries
WSJ	173,252	Disks 1-2	151-200
AP	242,918	Disks 1-3	51-150
ROBUST	528,155	Disks 4-5 (-CR)	301-450, 600-700
WT10G	1,692,096	WT10g	451-550
GOV2	25,205,179	GOV2	701-850
CW09B CW09BF	50,220,423	ClueWeb09 Category B	1-200
CW09A CW09AF	503,903,810	ClueWeb09 Category A	1-200

Table 2: TREC data used to evaluate the quality of prediction over queries.

Titles of TREC topics served as queries. We applied Krovetz stemming via the Indri toolkit[8], which was used for experiments. Stopwords on the INQUERY list [1] were removed from queries, but not from documents.

We used our approach to predict the effectiveness of the query likelihood (QL) retrieval model [48] which is often used in reports on query-performance prediction [15, 54, 24, 53, 45, 46, 29]. Dirichlet-smoothed unigram language models are used with the smoothing parameter set to 1000. Log query-likelihood values are used for retrieval scores.

The similarity between documents d_i and d_j, used by the cohesion feature function, is defined as the exponent of the negative cross entropy between the unsmoothed language model of d_i and the Dirichlet-smoothed (with the smoothing parameter set to 1000) language model of d_j.

Except for the pre-retrieval predictors, *all* the predictors defined as feature functions in LTRoq analyze the n most highly ranked documents in the retrieved list. To learn which values of n are better than others for each of these predictors we do the following. For each feature function $f_x(x)$ defined over a clique x, instead of using a single value of n we use a linear combination of feature functions, $\sum_n \lambda_{x;n} f_{x;n}(x)$; $f_{x;n}(x)$ is a feature function defined exactly as $f_x(x)$, but analyzes only the n highest ranked documents in the list. Each linear combination per feature function, as that just defined, is weighed by λ_x following Equation 8.

All the predictors that were used as feature functions in LTRoq serve as reference comparisons. As already mentioned, a few of these are state-of-the-art. We also use as a baseline the state-of-the-art Query Feedback (QF) post-retrieval predictor [55]. A relevance language model induced from the n most highly ranked documents in the retrieved list is used to rank the entire collection. The number of documents that are among the top ν_{QF} documents in both the original retrieved list and that retrieved by using the relevance model serves as the prediction value; ν_{QF} is a free parameter. QF was not used as a feature function in LTRoq because it has two free-parameters, n and ν_{QF}, and their effective values do not generalize well across queries [46].

The reference comparison predictors are based on a single value of n as is standard. This value is set via cross validation as described below. Note that in contrast, LTRoq integrates instantiations of the same predictor with various values of n as feature functions.

Following common practice [11], prediction over queries quality is measured by the Pearson correlation between the values assigned to queries by a predictor and the actual average precision (AP@1000) computed for these queries using TREC's relevance judgments.

The weights associated with feature functions in LTRoq are learned in two separate phases. We use SVM^{rank} [26], applied with default free-parameter values, to learn weights in each phase using the same query train set. Queries are ranked for SVM^{rank} by the AP(@1000) attained using the QL retrieval method.

In the first phase, we learn *independently* for each feature function $f_x(x)$ the values of the $\lambda_{x;n}$ weights in the linear combination it forms: $\sum_n \lambda_{x;n} f_{x;n}(x)$. As the pre-retrieval predictors are independent of n, we treat them as a separate linear combination of feature functions. The weight associated with each feature function that corresponds to a pre-retrieval predictor is learned, independently, in the first phase. In the second phase, we learn the values of the λ_x weights of the linear combinations of the feature functions, including that of the pre-retrieval predictors.

To learn and test the weights of the feature functions in LTRoq, and the value of n for the reference comparison predictors, we randomly split the queries per each experimental setting into two equal-sized sets and use two-fold cross validation. The train set is used to learn the feature-functions' weights in LTRoq in two phases as described above. The

[8] www.lemurproject.org/indri

	WSJ	AP	ROBUST	WT10G	GOV2	CW09B	CW09BF	CW09A	CW09AF
SumSCQ	−.062⋆	−.182⋆	.072⋆	.141⋆	.267⋆	.229⋆	.230⋆	.455	.314⋆
AvgSCQ	.391⋆	.469⋆	.238⋆	.320⋆	.311⋆	.243⋆	.220⋆	.151⋆	.198⋆
MaxSCQ	.409⋆	.338⋆	.341⋆	.406⋆	.357⋆	.362⋆	.331⋆	.364⋆	.317⋆
SumVAR	.194⋆	.153⋆	.279⋆	.256⋆	.360⋆	.275⋆	.267⋆	.455	.313⋆
AvgVAR	.483⋆	.645⋆	.407⋆	.312⋆	.361⋆	.255⋆	.224⋆	.170⋆	.190⋆
MaxVAR	.395⋆	.497⋆	.424⋆	.406⋆	.384⋆	.358⋆	.322⋆	.360⋆	.262⋆
SumIDF	.015⋆	−.103⋆	.267⋆	.195⋆	.318⋆	.278⋆	.268⋆	**.493**⋆	.345⋆
AvgIDF	.271⋆	.415⋆	.394⋆	.180⋆	.268⋆	.237⋆	.211⋆	.173⋆	.207⋆
MaxIDF	.147⋆	.207⋆	.429⋆	.233⋆	.287⋆	.330⋆	.293⋆	.380⋆	.306⋆
entropy	−.160⋆	−.185⋆	−.120⋆	−.171⋆	.248⋆	.419⋆	.492⋆	.388⋆	.401⋆
sw1	−.010⋆	−.053⋆	−.049⋆	−.076⋆	.307⋆	.435⋆	.409⋆	.430⋆	.360⋆
sw2	−.130⋆	−.045⋆	−.110⋆	−.026⋆	.318⋆	.419⋆	.439⋆	.281⋆	.403
cohesion	.140⋆	.279⋆	−.004⋆	−.036⋆	.232⋆	−.110⋆	−.040⋆	−.140⋆	−.171⋆
Clarity	.551⋆	.569⋆	.371⋆	.153⋆	.104⋆	−.239⋆	−.197⋆	−.363⋆	−.267⋆
ImpClarity	.309⋆	.565⋆	.453⋆	.256⋆	.352⋆	.166⋆	.091⋆	.030⋆	−.078⋆
WIG	.620⋆	.542⋆	.493⋆	.399⋆	.466⋆	.281⋆	.319⋆	.256⋆	.328⋆
NQC	.654⋆	.526⋆	.496⋆	.405⋆	.336⋆	.234⋆	.406⋆	.067⋆	.274⋆
UEF(Clarity)	.611⋆	.613⋆	.537⋆	.327⋆	.437⋆	−.023⋆	.133⋆	−.097⋆	−.004⋆
UEF(ImpClarity)	.543⋆	.636⋆	**.563**	.421⋆	.537⋆	.265⋆	.382⋆	.028⋆	.217⋆
UEF(WIG)	.558⋆	.575⋆	.538⋆	.413⋆	.502⋆	.361⋆	.466⋆	.063⋆	.375⋆
UEF(NQC)	.639⋆	.552⋆	.510⋆	**.435**⋆	.417⋆	.272⋆	.458⋆	.069⋆	.253⋆
QF	.462⋆	.495⋆	.457⋆	.378⋆	.374⋆	.221⋆	.273⋆	.068⋆	.166⋆
LTRoq	**.695**	**.692**	.557	.346	**.570**	**.512**	**.535**	.457	**.422**

Table 3: Prediction (over queries) quality of LTRoq. '⋆' marks a statistically significant difference with LTRoq. The best result per experimental setting is boldfaced.

value of n for the reference comparisons is learned by optimizing Pearson's correlation over the train set. The evaluation score for a split of the query set is the average prediction quality over the two test sets defined by the split. We repeat this procedure 30 times, and report the average prediction quality over the 30 splits (i.e., Pearson's correlation). For fairness of comparison, the pre-retrieval predictors which serve as reference comparisons are evaluated using the same splits, although they do not incorporate free parameters. Statistically significant differences of prediction quality are determined using the two-tailed paired t-test with $p < 0.05$ computed over the query set splits.

To construct a relevance language model [30], which is used in Clarity, ImpClarity, UEF and QF, we used unsmoothed document language models and set the number of terms to 100 [46]; for ImpClarity we set $t = 1$. The value of n in the prediction methods that rely on it was selected from $\{5, 10, 25, 50, 100, 250, 500, 1000\}$. For QF, we set ν_{QF} to a value in $\{5, 10, 25, 50, 100\}$.

4.1.2 Experimental results

The prediction quality of the various predictors is presented in Table 3. Our first observation based on Table 3 is that, in general, the post-retrieval predictors yield prediction quality that surpasses that of the pre-retrieval predictors. A case in point, the most effective pre-retrieval predictor in an experimental setting is outperformed by the most effective post-retrieval predictor for seven out of the nine settings. The inferiority of the pre-retrieval predictors could be attributed to the fact that they are independent of the retrieved list that is evaluated.

Another observation that we make based on Table 3 is that in most cases for the ClueWeb09 settings the pre-retrieval predictors are more effective when used to predict the effectiveness of the retrieved list before suspected spam documents were filtered out, i.e., for CW09B and CW09A, rather than after they were removed (CW09BF and CW09AF). On the other hand, post-retrieval predictors are more effective when predicting the effectiveness of the retrieved list after spam documents were filtered out. This finding could potentially be attributed to the fact that post-retrieval predictors analyze the retrieved list, while pre-retrieval predictors utilize only corpus-based query-term statistics which does not change by removing spam documents (only) from the list.

Most importantly, we see in Table 3 that for six out of the nine experimental settings, the best prediction quality is attained by LTRoq. Furthermore, LTRoq outperforms each of the reference comparisons (often to a statistically significant degree) in a vast majority of the experimental settings. LTRoq can be (sometimes statistically significantly) outperformed by some of the predictors for ROBUST, WT10G and CW09A. However, none of the predictors statistically significantly outperforms LTRoq for more than two settings. As UEF, NQC, WIG and QF are state-of-the-art methods, we conclude that LTRoq is the most effective prediction-over-queries method reported in the literature.

Feature function analysis. We next study the importance of the different feature functions used in LTRoq. To this end, we present in Table 4 for each experimental setting the five feature functions assigned with the highest λ_x values by SVM^{rank} in the second phase of training. Recall that the weights assigned to feature functions in LTRoq are learned in two phases. The weights of the pre-retrieval predictors are set in the first phase of training; the weight of the linear combination of these predictors is set in the second phase. Hence, the pre-retrieval predictors are treated here as a single group, denoted **Pre**.

We can see in Table 4 that, with the exception of WSJ and GOV2, the pre-retrieval predictors are always among the top-5 feature functions. In contrast, when not integrated in LTRoq, pre-retrieval predictors are often outperformed by post-retrieval predictors as shown in Table 3.

Another observation that we make based on Table 4 is that for the Web settings (WT10G, GOV2 and the four ClueWeb09 settings) at least one of the feature functions defined over the x_L clique is always among the top-5 feature functions; for the newswire collections, these feature functions are never among the top-5 feature functions. This finding attests to the merits of using feature functions that

WSJ	AP	ROBUST	WT10G	GOV2	CW09B	CW09BF	CW09A	CW09AF
UEF(NQC)	UEF(Clarity)	NQC	Pre	sw2	sw1	sw1	Pre	sw1
NQC	UEF(ImpClarity)	UEF(ImpClarity)	sw1	sw1	sw2	sw2	sw1	UEF(WIG)
UEF(Clarity)	Clarity	Pre	UEF(NQC)	UEF(WIG)	Pre	UEF(WIG)	cohesion	entropy
UEF(WIG)	UEF(NQC)	UEF(NQC)	UEF(ImpClarity)	ImpClarity	UEF(ImpClarity)	Pre	UEF(WIG)	Pre
UEF(ImpClarity)	Pre	ImpClarity	UEF(WIG)	UEF(NQC)	cohesion	NQC	NQC	sw2

Table 4: The top-5 feature functions used by LTRoq for prediction over queries. "Pre": pre-retrieval predictors.

analyze query-independent properties of the retrieved list for prediction over queries in Web collections, as was also demonstrated elsewhere [40].

We can also see that feature functions associated with all four types of cliques used by LTRoq are represented in the top-5 feature functions. As just noted, the feature functions defined over the x_L clique are among the top-5 feature functions for all Web settings. The pre-retrieval predictors which are defined over the x_{QC} clique are almost always among the top-5 feature functions. UEF, which is defined over the x_{QLC} clique, is always among the top-5 feature functions when instantiated with either WIG or ImpClarity. The least represented clique in the top-5 feature functions is x_{QL} which has representatives (Clarity or ImpClarity) for only three out of the nine settings.

4.2 Prediction over retrieved lists

We next study the effectiveness of utilizing the feature functions defined in Section 3.5.1, and which were used by LTRoq to rank queries, for the prediction over retrieved lists task. To this end, we use a learning-to-rank approach, denoted **LTRol**, for ranking lists. Recall that these feature functions are previously proposed predictors over queries. Our goal in using these feature functions here is to empirically examine our claim from Section 3.4. That is, these predictors, which are essentially query ranking functions, should not be expected to be effective for ranking retrieved lists (i.e., for the prediction over retrieved lists task).

4.2.1 Experimental setup

Let $\{l_i\}$ be a set of lists retrieved in response to a query q using some retrieval methods. $S(d;q,l)$ is the (sum-normalized) retrieval score of document d in l ($\in \{l_i\}$); $S(d;q,l) \stackrel{def}{=} 0$ if d is not in l. Linear fusion methods [3] assign d with the score $\sum_{l \in \{l_i\}} \alpha_l S(d;q,l)$, where α_l is a (non-negative) weight assigned to list l. The prediction over retrieved lists task that we focus on here is learning the α_l weights. These weights should reflect the effectiveness of the lists with respect to q.

Runs submitted to TREC serve as the results of different retrieval methods. Details of the TREC benchmarks are provided in Table 5. We randomly sample 5 runs out of all the runs submitted to a track; the runs sampling procedure is repeated 30 times. The 100 most highly ranked documents for a query in a run serve as a retrieved list to be fused.[9] We use the mean average precision at cutoff 100 (MAP), and the precision of the top-5 documents (p@5), averaged over the 30 samples, as evaluation measures. Statistically significant differences of retrieval performance are computed with respect to the average performance over the 30 samples.

The pre-retrieval predictors are not used here, neither when applied alone nor when integrated in LTRol, as they are independent of a retrieved list and LTRol uses a linear combination of predictors. The remaining feature functions that are used in LTRol are computed for each retrieved list. While in Section 4.1 these feature functions were used to rank queries, here we use them to weigh the lists.

[9]Fusion methods were shown to be most effective when fusing relatively short lists [47, 7].

TREC	track	# of docs	data	queries
TREC7	Ad hoc	528,155	Disks 4-5 (-CR)	351-400
TREC8				401-450
TREC9	Web	1,692,096	WT10g	451-500
TREC18	Web	50,220,423	ClueWeb09 Category B	1-50
TREC19				51-100
TREC20				101-150
TREC21				151-200

Table 5: Data used for the evaluation of fusion-based retrieval effectiveness when using prediction over retrieved lists to weigh lists.

Since our focus here is on ranking lists rather than queries, the query-length and corpus normalization factors used in WIG and NQC to ensure across query compatibility need not be computed [46]. Thus, WIG is the mean retrieval score in a retrieved list; NQC is the standard deviation. Instead of using QL retrieval scores to weigh documents when constructing the relevance model in Clarity, ImpClarity and UEF, document retrieval scores in the lists are used [46].

For reference comparison we use several state-of-the-art predictors *over queries* as list weights, namely WIG, NQC, UEF(WIG) and UEF(NQC). These were described in Section 3.5.1. We also report the performance of using the following three list-weighting schemes: (i) optimal weighting, **OPT**, where the weight assigned to a list is its actual AP(@100); (ii) **UNI**, where all the lists are uniformly weighted; this amounts to using the CombSum linear fusion method [20]; and, (iii) cross validation, **CV**, where the weight assigned to a retrieved list for queries in a test set is the MAP computed based on the queries in the train set for the same retrieval method (i.e., the same run). CV is based on the premise that past performance of a retrieval method (i.e., over train queries) indicates its performance with new queries. In contrast, *all* other predictors that we consider (except for UNI) weigh a list by directly estimating its effectiveness.

The number of the highest ranked documents in a retrieved list (n) considered in the feature functions and reference comparisons, and the weights of the feature functions in LTRol, are set using 5-fold cross validation; query IDs are used to create the folds. The weights in LTRol are learned using SVM^{rank} [26] in two phases as was the case for LTRoq[10]. AP@100 is used in the learning phase to rank the lists that are fused in the training set. Since each of the lists

[10]We apply min-max normalization to the α_l weights which are assigned by SVM^{rank} to the retrieved lists to avoid using negative weights.

		WorstRun	MedianRun	BestRun	OPT	UNI	CV	WIG	NQC	UEF(WIG)	UEF(NQC)	LTRol
TREC7	MAP	8.7^{u}	15.4^{u}_{c}	21.2	25.2^{u}_{c}	20.2_{c}	$\mathbf{21.4}^{u}$	20.5^{u}_{c}	18.9^{u}	20.6^{u}_{c}	19.0^{u}	20.6_{c}
	p@5	31.7^{u}_{c}	45.8^{u}_{c}	59.1^{u}_{c}	65.2^{u}_{c}	54.4_{c}	56.7^{u}	54.1_{c}	50.7^{u}_{c}	54.1_{c}	50.8^{u}_{c}	$\mathbf{57.3}^{u}$
TREC8	MAP	7.8^{u}_{c}	20.2^{u}_{c}	27.2^{u}	30.5^{u}_{c}	25.8_{c}	$\mathbf{27.6}^{u}$	26.6^{u}_{c}	26.3_{c}	26.6^{u}_{c}	26.5^{u}_{c}	26.5^{u}_{c}
	p@5	24.1^{u}_{c}	50.9^{u}_{c}	64.9^{u}	68.5^{u}_{c}	60.9_{c}	$\mathbf{63.2}^{u}$	61.3_{c}	61.1_{c}	61.4^{u}_{c}	61.4_{c}	62.5^{u}
TREC9	MAP	4.5^{u}_{c}	13.0^{u}_{c}	19.7^{u}	23.4^{u}_{c}	17.1_{c}	$\mathbf{19.0}^{u}$	17.1_{c}	14.6^{u}_{c}	17.1_{c}	14.8^{u}_{c}	17.1_{c}
	p@5	12.6^{u}_{c}	25.6^{u}_{c}	38.7^{u}	45.5^{u}_{c}	34.8_{c}	$\mathbf{38.3}^{u}$	34.5^{u}_{c}	29.6^{u}_{c}	34.0^{u}_{c}	30.2^{u}_{c}	34.1_{c}
TREC18	MAP	5.6^{u}_{c}	10.4^{u}_{c}	14.6	18.2^{u}_{c}	14.3_{c}	$\mathbf{14.9}^{u}$	14.2_{c}	12.4^{u}_{c}	14.2_{c}	12.3^{u}_{c}	13.9^{u}_{c}
	p@5	21.9^{u}_{c}	31.5^{u}_{c}	36.1	43.7^{u}_{c}	36.5	36.6	36.3	32.5^{u}_{c}	**36.9**	32.5^{u}_{c}	35.5
TREC19	MAP	4.5^{u}_{c}	11.2^{u}_{c}	16.9^{u}	20.7^{u}_{c}	14.8_{c}	$\mathbf{16.9}^{u}$	14.7^{u}_{c}	8.5^{u}_{c}	14.6_{c}	8.5^{u}_{c}	16.5^{u}
	p@5	16.0^{u}_{c}	27.3_{c}	38.8^{u}	42.4^{u}_{c}	28.6_{c}	34.4^{u}	28.1^{u}_{c}	21.5^{u}_{c}	28.6_{c}	22.0^{u}_{c}	$\mathbf{36.4}^{u}_{c}$
TREC20	MAP	7.2^{u}_{c}	13.8^{u}_{c}	20.3	28.4^{u}_{c}	21.1_{c}	$\mathbf{22.0}^{u}$	21.2_{c}	17.5^{u}_{c}	20.8_{c}	17.5^{u}_{c}	19.9^{u}_{c}
	p@5	19.7^{u}_{c}	29.9^{u}_{c}	36.7	43.5^{u}_{c}	37.7_{c}	$\mathbf{38.5}^{u}$	37.7_{c}	35.2^{u}_{c}	37.5_{c}	35.2^{u}_{c}	37.2_{c}
TREC21	MAP	9.0^{u}_{c}	14.6^{u}_{c}	20.8	26.4^{u}_{c}	21.6_{c}	$\mathbf{22.4}^{u}$	21.8	18.7^{u}_{c}	21.7_{c}	18.9^{u}_{c}	20.8_{c}
	p@5	30.0^{u}_{c}	31.6^{u}_{c}	39.1	45.5^{u}_{c}	37.7_{c}	38.7^{u}	37.2_{c}	34.2^{u}_{c}	37.4_{c}	34.7^{u}_{c}	$\mathbf{42.3}^{u}_{c}$

Table 6: Retrieval effectiveness of using LTRol to weigh lists for linear fusion. WorstRun, MedianRun and BestRun: average performance of the worst, median and best run, among the five fused, respectively. 'u' and 'c' mark statistically significant differences with UNI and CV, respectively. The best result of a prediction method per experimental setting and evaluation metric is boldfaced.

to be fused is composed of 100 documents, the value of n is selected from $\{5, 10, 25, 50, 100\}$. All other implementation details are the same as those described in Section 4.1.1.

4.2.2 Experimental results

The results are presented in Table 6. **WorstRun**, **MedianRun** and **BestRun** denote the average performance, over the 30 samples, of the worst, median and best run among the 5 runs in a sample, respectively. The substantial performance differences among the three, along with the very high OPT performance numbers, attest to the substantial merits of assigning effective weights to the lists when fusing them.

We can see that LTRol is more effective than NQC and UEF(NQC) in most relevant comparisons (track × evaluation metric). LTRol is also more effective than WIG and UEF(WIG) in about half of the relevant comparisons. Furthermore, LTRol is the only method among those using predictors over queries which in most relevant comparisons outperforms UNI (i.e., uniform weighting of lists). Nonetheless, LTRol is outperformed in most cases by CV. This means that weighing a retrieved list based on the *past* performance of the retrieved method that produces the list is much more effective than applying prediction-over-queries methods to the list itself to set its weight. Thus, we get empirical support to the claim from Section 3.4 that using predictors over queries to rank retrieved lists is not highly effective.[11]

5. SUMMARY

We framed the query-performance prediction task in the most general probabilistic terms. This gave rise to novel connections between tasks, and methods used to address them, in federated search, fusion-based retrieval, and previous work on query-performance prediction (i.e., prediction over queries). We formally argued that successful prediction directly implies knowledge of how to perform effective retrieval. We also argued why using previously proposed query-performance predictors was not shown to improve retrieval effectiveness. These are query ranking methods that are suited for specific retrieval methods. In addition, we devised a learning-to-rank approach for predicting performance over queries. The resultant prediction quality substantially transcends that of state-of-the-art predictors. We used our approach to show that predictors over queries are not very effective for prediction over retrieved lists.

[11]Learning-to-rank methods were used to fuse lists [43] and to rank them [5]. Both approaches use features specific to, and shared by, the rankers employed. In the TREC setting here, the features used by the rankers are not necessarily known, nor shared. We used predictors over queries that can be applied to any retrieved list.

Acknowledgments We thank the reviewers for their comments and Yun Zhou for a discussion about the connections between prediction tasks. This work was supported by and carried out at the Technion-Microsoft Electronic Commerce Research Center. This work was also supported in part by Microsoft Research through its Ph.D. Scholarship Program.

6. REFERENCES

[1] J. Allan, M. E. Connell, W. B. Croft, F.-F. Feng, D. Fisher, and X. Li. INQUERY and TREC-9. In *Proc. of TREC-9*, pages 551–562, 2000.

[2] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In *Proc. of ECIR*, pages 127–137, 2004.

[3] C. C. V. ant Garrison W. Cottrell. Fusion via linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[4] J. A. Aslam and V. Pavlu. Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions. In *Proc. of ECIR*, pages 198–209, 2007.

[5] N. Balasubramanian and J. Allan. Learning to select rankers. In *Proc. of SIGIR*, pages 855–856, 2010.

[6] N. Balasubramanian, G. Kumaran, and V. R. Carvalho. Predicting query performance on the web. In *Proc. of SIGIR*, pages 785–786, 2010.

[7] S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, D. A. Grossman, and N. Goharian. Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *Proc. of SAC*, pages 823–827, 2003.

[8] M. Bendersky, W. B. Croft, and Y. Diao. Quality-biased ranking of web documents. In *Proc. of WSDM*, pages 95–104, 2011.

[9] Y. Bernstein, B. Billerbeck, S. Garcia, N. Lester, F. Scholer, and J. Zobel. RMIT university at trec 2005: Terabyte and robust track. In *Proc. of TREC-14*, 2005.

[10] J. Callan. Distributed information retrieval. In W. Croft, editor, *Advances in information retrieval*, chapter 5, pages 127–150. Kluwer Academic Publishers, 2000.

[11] D. Carmel and E. Yom-Tov. *Estimating the Query Difficulty for Information Retrieval*. Synthesis lectures on information concepts, retrieval, and services. Morgan & Claypool, 2010.

[12] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *Proc. of SIGIR*, pages 390–397, 2006.

[13] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Informaltiom Retrieval Journal*, 14(5):441–465, 2011.

[14] W. B. Croft. Combining approaches to information retrieval. In W. B. Croft, editor, *Advances in information retrieval*, chapter 1, pages 1–36. Kluwer Academic Publishers, 2000.
[15] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of SIGIR*, pages 299–306, 2002.
[16] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. A language modeling framework for selective query expansion. Technical Report IR-338, Center for Intelligent Information Retrieval, University of Massachusetts, 2004.
[17] R. Cummins. Predicting query performance directly from score distributions. In *Proc. of AIRS*, pages 315–326, 2011.
[18] R. Cummins, J. M. Jose, and C. O'Riordan. Improved query performance prediction using standard deviation. In *Proc. of SIGIR*, pages 1089–1090, 2011.
[19] F. Diaz. Performance prediction using spatial autocorrelation. In *Proc. of SIGIR*, pages 583–590, 2007.
[20] E. A. Fox and J. A. Shaw. Combination of multiple searches. In *Proc. of TREC-2*, 1994.
[21] C. Hauff and L. Azzopardi. When is query performance prediction effective? In *Proc. of SIGIR*, pages 829–830, 2009.
[22] C. Hauff, L. Azzopardi, and D. Hiemstra. The combination and evaluation of query performance prediction methods. In *Proc. of ECIR*, pages 301–312, 2009.
[23] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *Proc. of CIKM*, pages 1419–1420, 2008.
[24] C. Hauff, V. Murdock, and R. A. Baeza-Yates. Improved query difficulty prediction for the web. In *Proc. of CIKM*, pages 439–448, 2008.
[25] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *Proc. of SPIRE*, pages 43–54, 2004.
[26] T. Joachims. Training linear svms in linear time. In *Proc. of KDD*, pages 217–226, 2006.
[27] O. Kurland, F. Raiber, and A. Shtok. Query-performance prediction and cluster ranking: Two sides of the same coin. In *Proc. of CIKM*, pages 2459–2462, 2012.
[28] O. Kurland, A. Shtok, D. Carmel, and S. Hummel. A unified framework for post-retrieval query-performance prediction. In *Proc. of ICTIR*, pages 15–26, 2011.
[29] O. Kurland, A. Shtok, S. Hummel, F. Raiber, D. Carmel, and O. Rom. Back to the roots: a probabilistic framework for query-performance prediction. In *Proc. of CIKM*, pages 823–832, 2012.
[30] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Proc. of SIGIR*, pages 120–127, 2001.
[31] D. Lillis, F. Toolan, R. W. Collier, and J. Dunnion. Probfuse: a probabilistic approach to data fusion. In *Proc. of SIGIR*, pages 139–146, 2006.
[32] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.
[33] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Proc. of SIGIR*, pages 186–193, 2004.
[34] X. Liu and W. B. Croft. Experiments on retrieval of optimal clusters. Technical Report IR-478, Center for Intelligent Information Retrieval (CIIR), University of Massachusetts, 2006.
[35] C. Macdonald, R. L. T. Santos, and I. Ounis. On the usefulness of query features for learning to rank. In *Proc. of CIKM*, pages 2559–2562, 2012.
[36] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. of SIGIR*, pages 472–479, 2005.
[37] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty. In *ACM SIGIR 2005 Workshop on Predicting Query Difficulty - Methods and Applications*, 2005.
[38] J. Pérez-Iglesias and L. Araujo. Standard deviation as a query hardness estimator. In *Proc. of SPIRE*, pages 207–212, 2010.
[39] F. Raiber and O. Kurland. Ranking document clusters using markov random fields. In *Proc. of SIGIR*, pages 333–342, 2013.
[40] F. Raiber and O. Kurland. Using document-quality measures to predict web-search effectiveness. In *Proc. of ECIR*, pages 134–145, 2013.
[41] F. Scholer and S. Garcia. A case for improved evaluation of query difficulty prediction. In *Proc. of SIGIR*, pages 640–641, 2009.
[42] F. Scholer, H. E. Williams, and A. Turpin. Query association surrogates for web search. *JASIST*, 55(7):637–650, 2004.
[43] D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. Lambdamerge: merging the results of query reformulations. In *Proc. of WSDM*, pages 795–804, 2011.
[44] M. Shokouhi and L. Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, 2011.
[45] A. Shtok, O. Kurland, and D. Carmel. Using statistical decision theory and relevance models for query-performance prediction. In *Proc. of SIGIR*, 2010.
[46] A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits. Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems*, 30(2):11, 2012.
[47] I. Soboroff, C. K. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proc. of SIGIR*, pages 66–73, 2001.
[48] F. Song and W. B. Croft. A general language model for information retrieval (poster abstract). In *Proc. of SIGIR*, pages 279–280, 1999.
[49] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Information Processing and Management*, 36(6):779–808, 2000.
[50] S. Tomlinson. Robust, Web and Terabyte Retrieval with Hummingbird Search Server at TREC 2004. In *Proc. of TREC-13*, 2004.
[51] V. Vinay, I. J. Cox, N. Milic-Frayling, and K. R. Wood. On ranking the effectiveness of searches. In *Proc. of SIGIR*, pages 398–404, 2006.
[52] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proc. of SIGIR*, pages 512–519, 2005.
[53] Y. Zhao, F. Scholer, and Y. Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proc. of ECIR*, pages 52–64, 2008.
[54] Y. Zhou and B. Croft. Ranking robustness: a novel framework to predict query performance. In *Proc. of CIKM*, pages 567–574, 2006.
[55] Y. Zhou and B. Croft. Query performance prediction in web search environments. In *Proc. of SIGIR*, pages 543–550, 2007.

APPENDIX

A. POST-RETRIEVAL PREDICTION OVER QUERIES

We argue that post-retrieval prediction over queries methods (henceforth prediction methods or predictors) were coupled with, and are hence effective for, specific retrieval methods; that is, for ranking queries by the presumed effectiveness of using these retrieval methods for them.

There are predictors that explicitly rely on features used by the retrieval method [6]. This reliance is implicit for other prediction methods as we discuss next.

As described in Section 2, there are three classes of post-retrieval predictors. The Clarity predictor [15] was shown to be ineffective for a variety of retrieval methods [41].

Predictors that analyse the retrieval scores distribution [50, 9, 19, 55, 6, 38, 17, 18, 46] are by definition dependent on the retrieval method as noted in some work [55, 46]. In fact, almost all of these predictors are suited for, and were evaluated with, retrieval methods that use *only* document-query surface-level similarities to rank documents.

Query feedback (QF) [55] and UEF [45] are among the most effective post-retrieval predictors that do not analyze retrieval scores. Both rely on the premise that retrieval using pseudo-feedback-based query expansion is more effective than that at hand. Indeed, in both reports [55, 45], the retrieval method used for evaluation was a language-model-based approach with no query expansion. Comparison with query-expansion-based retrieval, or some other effective retrieval, need not necessarily indicate the effectiveness of retrieval methods that are much more effective [47].