Robust Standard Deviation Estimation for Query Performance Prediction

Haggai Roitman, Shai Erera, Bar Weiner
IBM Research - Haifa
Haifa, Israel 31905
haggai,shaie,barw@il.ibm.com

ABSTRACT

We derive a robust standard deviation estimator for post-retrieval query performance prediction. To this end, we propose a novel bootstrap sampling approach which is inspired by user search behavior. Using an evaluation with several TREC benchmarks and a comparison with several different types of baselines, we demonstrate that, overall, our estimator results in an enhanced query performance prediction.

1 BACKGROUND

We address the problem of post-retrieval query performance prediction [2] (QPP). A post-retrieval QPP method predicts the query's performance based on the quality of its retrieved result list [2].

A common approach utilized by many state-of-the art post-retrieval QPP methods is the analysis of retrieval scores [6, 8, 10, 13, 15, 18]. Specifically, the **standard deviation** of retrieval scores has been employed as a strong indicator of query performance [6, 8, 13, 15]. Higher standard deviation usually attests to lower chance of query-drift [13] or higher content diversity [3], which in turn, is assumed to result in a better query performance. Various alternative methods have been proposed for estimating the standard deviation for QPP [6, 8, 13, 15]. Each of these methods mostly differs in the way documents from the query's corresponding result list are selected for the estimation.

Inspired by [9, 12], we propose a new standard deviation estimator that utilizes several document result lists as reference lists for enhanced QPP. Each such list is sampled from the original retrieved result list. This in comparison to most previous works which directly estimate the standard deviation using only the original retrieved result list [6, 8, 13, 15]. To this end, various document samples are obtained using a novel bootstrap sampling approach. The proposed sampling approach simulates a random user that "browses" the retrieved result list and "clicks" on the documents that should be included in a given sample. Various samples are further weighted based on their presumed quality [9, 12]. Noting that score divergence may be query-dependent [6, 13, 15], we further normalize the bootstrap estimate to ensure inter-query compatibility. We show that, the *Normalized Query Commitment* (NQC) method [13] can be derived as a private instance of our estimator.

Using an evaluation with several TREC benchmarks and a comparison with several different types of baselines (including two variants of the Bagging method [1]), we demonstrate that, overall, our estimator results in an enhanced query performance prediction.

2 ESTIMATION APPROACH

For a given query q and corpus C, let D denote the result list of the top-k documents $d \in C$ with the highest retrieval scores $s(d|q)$.

We focus on post-retrieval QPP [2]. Therefore, our goal is to estimate *the likelihood of finding relevant information for q in D* [2]. Similar to many previous works [6, 8, 10, 13, 15, 18], we estimate such likelihood by analyzing the retrieval scores $s(d|q)$ of the documents in D. Specifically, we focus on the (retrieval scores) standard deviation, hereinafter denoted $\sigma_{s|q}$, as the indicator of query performance. Higher standard deviation usually attests to lower chance of query-drift [13] or higher content diversity [3], which in turn, is assumed to result in a better query performance.

2.1 Robust Standard Deviation estimator

Our goal is to estimate $\sigma_{s|q}$ as accurately as possible. To this end, we propose a novel bootstrap sampling approach that estimates $\sigma_{s|q}$ using $N \geq 1$ samples $D_1, D_2, \ldots, D_N$; each sample satisfies $D_j \subseteq D$ and contains $|D_j| = l \leq k$ documents.

Our proposed bootstrap estimator, hereinafter named **RSD**[1], is calculated as follows:

$$\hat{\sigma}_{s|q} \overset{def}{=} \sqrt{\sum_{j=1}^{N} \omega(D_j) \cdot \widehat{var}(s|q, D_j)} \quad (1)$$

$\widehat{var}(s|q, D_j) \overset{def}{=} \frac{1}{l-1} \sum\limits_{d \in D_j} \left(s(d|q) - \hat{\mu}_{D_j}\right)^2$ denotes the (unbiased) variance estimate of sample D_j's document scores, where $\hat{\mu}_{D_j} \overset{def}{=} \frac{1}{l} \sum\limits_{d \in D_j} s(d|q)$ denotes D_j's mean score. $\omega(D_j)$ is a non-negative real weight, which denotes the relative importance of sample D_j.

2.2 Bootstrap sampling approach

Each $\widehat{var}(s|q, D_j)$ is estimated based on a random sample of documents $D_j \subseteq D$ obtained by a new proposed **ranked-biased**, **WOR** (without replacement) and **round-robin** bootstrap sampling approach. Using this approach, each new sample D_j is obtained by simulating the behavior of a "random user" who "browses" the documents in D, retrieved in response to the user's query q.

ICTIR '17, October 01–04, 2017, Amsterdam, Netherlands.
© 2017 ACM. ISBN 978-1-4503-4490-6/17/10…$15.00
DOI: https://doi.org/10.1145/3121050.3121087

[1] RSD stands for "Robust Standard Deviation".

The random user is assumed to scan the result list D from top to bottom. On each document, its likelihood of being included in D_j is assumed to be relative to its likelihood of being "clicked" by the random user. Similar to real-world search settings, such random clicks are assumed to be rank-biased [4]. Therefore, the higher document d's ($\in D$) rank is, the higher its chance of being clicked by the random user (hence **ranked-biased** sampling). Let r_d denote document d's ($\in D$) rank ($1 \leq r_d \leq k$) and let $u \sim U[0,1]$. Using an acceptance-rejection approach, d is included ("clicked") in sample D_j, if the following condition[2] is satisfied: $\sum_{r=r_d}^{k} p(r) \geq u$, where $p(r) \overset{def}{=} \frac{2(k+1-r)}{k(k+1)}$ $(1 \leq r \leq k)$ denotes the rank distribution.

We also assume that, the random user may click each document only once (hence a **WOR** sampling). Finally, for a given required sample size $|D_j| = l \leq k$, the random user is assumed to click exactly l documents in D. Therefore, whenever the random user has reached the bottom of the list and there are still documents to click, the random user continues her scan again from the top of the list (hence a **round-robin** sampling).

2.3 Sample weighting

We next suggest three variants of the RSD estimator, which was defined in Eq. 1, based on the sample weighting scheme $\omega(D_j)$ that is being employed. The first weighting scheme (denoted "uni") assumes that all samples have the same importance and assigns $\omega_{\mathsf{uni}}(D_j) \overset{def}{=} \frac{1}{N}$. Using the uni scheme we estimate the standard deviation according to the average of the samples' variances.

Inspired by [9, 12], we note that, each sample D_j can be treated as a *reference list* for prediction [12]. Hence, different samples can be weighted according to their own predicted query performance. Further following [12], as a second weighting scheme (denoted "sim"), we measure $\omega_{\mathsf{sim}}(D_j) \overset{def}{=} sim_{RBO(p)}(D_j, D)$ using the rank-biased overlap [16] (RBO) similarity measure. p is a free parameter set to 0.95, following [12]. Here, each sample D_j is assumed to be a pseudo effective (PE) reference list [12]; therefore, the higher the similarity between D and D_j, the more important D_j is [12].

We also propose a third weighting scheme inspired by the WIG [18] method[3] (denoted "wig"), which is calculated as: $\omega_{\mathsf{wig}}(D_j) \overset{def}{=} \frac{1}{l} \sum_{d \in D_j} (s(d|q) - s(C|q))$, where $s(C|q)$ denotes the corpus query likelihood. $s(C|q)$ can be estimated by treating the corpus as a single document. According to $\omega_{\mathsf{wig}}(D_j)$, a sample D_j whose document scores' deviate more from the corpus score (which acts as an ineffective reference document [18]), is assumed to contain better documents, and therefore, receives a higher weight[4].

Overall, using the three sample weighting schemes, we obtain three variants of our proposed RSD estimator, hereinafter denoted RSD[uni], RSD[sim] and RSD[wig], respectively.

2.4 Query-sensitive normalization

We note that, score divergence may be query-dependent [6, 13, 15]. Hence, following [6, 13, 15], to ensure inter-query compatibility, we further normalize our estimate of $\hat{\sigma}_{s|q}$ as follows:

$$\hat{\sigma}^{norm}_{s|q} \overset{def}{=} \frac{\hat{\sigma}_{s|q} \cdot nperp(q|R)}{|s(C)|}. \tag{2}$$

Here, similar to [13, 15], $|s(C)|$ denotes the (absolute) corpus query likelihood. Similar to WIG, such normalization utilizes the corpus as an ineffective reference document; the higher $|s(C)|$ is, the more difficult query q is assumed to be [13, 15].

We further introduce a second (and new) normalization term $nperp(q|R)$, which models to what extent q provides a correct representation of the (hidden) information need I_q [14]. Thus, the higher $nperp(q|R)$ is, the better q's performance is assumed to be. To this end, assuming that the relevance model [7] R induced from D approximates I_q, the representativeness of q (having n_q unique terms) given R is calculated according to the *normalized perplexity*: $nperp(q|R) \overset{def}{=} \frac{2^{H(q|R)}}{2^{\log n_q}}$. $H(q|R) \overset{def}{=} -\sum_{w \in q} p(w|R) \log p(w|R)$ is the weighted entropy of query q given R, where for each term $w \in q$ we assign the weight of 1 and 0 to the rest of the vocabulary [14]. Therefore, a query q that is more "anticipated" by the relevance model R, is assumed to provide a better representation of I_q.

3 EVALUATION

3.1 Datasets

Corpus	#documents	Queries	Disks
AP	242,918	51-150	1-3
TREC4	567,529	201-250	2-3
TREC5	524,929	251-300	2&4
ROBUST	528,155	301-450, 601-700	4&5-{CR}
WT10g	1,692,096	451-550	WT10g
GOV2	25,205,179	701-850	GOV2

Table 1: TREC benchmarks used for the evaluation.

Table 1 summarizes the TREC corpora and queries used for the evaluation. These benchmarks were used by many previous QPP works [2]. Titles of TREC topics were used as queries, except for the TREC4 benchmark, where no titles are available and topic descriptions were used instead. The Apache Lucene[5] open source search library was used for indexing and searching documents. Documents and queries were processed using Lucene's English text analysis (i.e., tokenization, Porter stemming, stopwords, etc.). As the underlying retrieval method, we used Lucene's Dirichlet-smoothed query-likelihood implementation with $\mu = 1000$ [17].

3.2 Baselines

We compared the three (normalized) variants of our proposed RSD estimator (i.e., RSD[uni], RSD[sim] and RSD[wig]) with several different types of baseline QPP methods.

[2] Note that, the first document in D is always "clicked" by the random user.

[3] WIG's original prediction is given by $\frac{\omega_{\mathsf{wig}}(D_j)}{\sqrt{|q|}}$. Yet, we do not divide in $\sqrt{|q|}$ since query q is fixed across all samples.

[4] To avoid negative weights we further take $\omega'_{\mathsf{wig}}(D_j) = \max\left\{0, \omega_{\mathsf{wig}}(D_j)\right\}$.

[5] http://lucene.apache.org

As a first line of baselines, we compared with the Clarity [5], WIG [18] and QF methods [18], which are commonly used as competitive post-retrieval QPP methods [2]. The Clarity [5] method estimates query performance proportionally to the divergence between the relevance language model [7] induced from D and that induced from C. The WIG method [18] estimates query performance according to the difference between the average retrieval score in D and that of C. The QF method predicts query performance according to the overlap between D and another list $D' \subseteq C$ (measured as $|D \cap D'|$), obtained by evaluating a new (weighted) query q' over C. q' is formulated from the top-n terms with the highest contribution to the KL-divergence between the relevance model induced from D and the background (corpus) model.

The next line of baselines we compared with were QPP methods that estimate the standard deviation in several alternative ways. This includes among others: σ_m which calculates the standard deviation using the top-m documents in D, hereinafter denoted $D^{[m]}$; using $m = 100$ following [8]; $\sigma_{\max}$ which takes the maximum standard deviation over all rank cutoffs [8]; $\sigma_{50\%}$ which considers only the document scores in D that are above the median score [6]; its extension $n(\sigma_{50\%}) \overset{def}{=} \frac{\sigma_{50\%}}{\sqrt{|q|}}$ [6]; and σ_k which adaptively decides on the rank cutoff for standard deviation calculation, with its tuning parameter set to $\lambda = 5$, following [6, 8].

We also compared with the NQC [13] method which can be derived as a private instance of RSD using D as a single "sample" and further setting $\omega(D) = 1$ and $nperp(q|R) = 1$. We also compared with the UEF[NQC] method [11], a more robust version of NQC, which multiplies the NQC value in the similarity $sim(D, \pi_D)$ (measured using Pearson's correlation on document scores [11]). π_D denotes the permutation of D obtained by re-ranking its documents using the relevance model induced from $D^{[m]}$. Finally, as another alternative that considers score "variance", we implemented the SMV method [15], whose prediction is calculated as follows: $\frac{1}{k|s(C)|} \sum_{d \in D} s(d) \left| \ln \frac{s(d)}{\hat{\mu}_D} \right|$.

In order to evaluate the effect of our proposed bootstrap sampling approach, we further implemented two alternative standard deviation estimators based on the bootstrap-aggregation (Bagging) method [1]. Similar to RSD, Bagging uses a bootstrap sampling approach to obtain several estimates of the variance, which are then averaged to obtain an aggregated estimate of the standard deviation[6]. Differently from RSD, the original Bagging's bootstrap sampling scheme, now denoted Bagging[org], is rank-oblivious and randomly selects documents with replacements (WR). We further implemented an extended approach, denoted Bagging[rank] which utilizes a rank-biased sampling approach similar to RSD (yet it still allows replacements). Similar to RSD, both Bagging variants were further normalized[7].

3.3 Setup

We predicted the performance of each query based on its top-1000 retrieved documents [2]. Following the common practice [2], we assessed prediction over queries quality according to the correlation between the predictor's values and the actual average precision (AP@1000) values calculated using TREC's relevance judgments. To this end, we report the Pearson's-ρ (P-ρ) and Kendall's-τ (K-τ) correlations which are the most common measures [2].

Most of the methods that we evaluated (including the RSD variants) required tuning some free parameters. Common to all methods is the free parameter $k \overset{def}{=} |D|$, which is the number of top scored documents (out of a total of 1000 retrieved documents) **to be used for the prediction**. To this end, for each method we selected $k \in \{5, 10, 20, 50, 100, 150, 200, 500, 1000\}$.

Next, some of the methods we evaluated required to tune additional parameters. For example, Clarity, QF, UEF[NQC] and $nperp(q|R)$, all utilize a relevance model [7] (RM1) that is induced from $D^{[m]}$, with $m \in \{1, 3, 5, \ldots, |D|\}$. Following [5, 11, 18], in all these methods, we further clipped the induced relevance model at the top-n terms cutoff, with $n \in \{5, 10, 20, 50, 100, 150, 200, 250\}$.

Finally, the number of bootstrap samples utilized by the RSD and Bagging variants was fixed to $N = 100$. Each documents sample size (**for all samples**) was further tuned as follows: $l \in \{30, 50, 100, 150, 200\}$.

Following [12, 13], training and testing of all methods was performed using a holdout (2-fold cross validation) approach. Accordingly, on each benchmark, we generated 30 random splits of the query set; each split had two folds. The first fold was used as the (query) train set, where parameters were tuned to maximize P-ρ. The second fold was kept untouched for testing. We recorded the average prediction quality (i.e., P-ρ and K-τ) over the 30 splits. Finally, we measured statistical significant differences of prediction quality using a two-tailed paired t-test with (Bonferroni corrected) $p < 0.05$ computed over all 30 splits.

3.4 Results

The results of our evaluation are summarized in Table 2. We first compare RSD to those baseline methods that predict query performance using only the (single) original retrieved result list D (either based on standard deviation or not). We then evaluate the impact of RSD's bootstrap sampling approach. Finally, we compare the three RSD variants to each other.

3.4.1 Original result list based prediction vs. RSD.

Comparing RSD side by side with the various single result list baselines (i.e., Clarity, WIG, QF and the various standard deviation estimation alternatives), we observe that, RSD provides a much better query performance prediction quality. Overall, compared to the **best alternative among these methods**, the best RSD variant (which was in all cases RSD[wig]) has provided an average improvement in prediction quality of 9.1%(±2.4%) and 8.5%(±3.9%) in P-ρ and K-τ, respectively. Such improvement was further statistically significant in most cases. This empirical result attests to the merits of using several reference lists for the standard deviation estimation based on our bootstrap sampling approach. Further notable is the improvement over NQC (including UEF[NQC]), which, as we have shown, can be derived as a private instance of our estimator.

[6] An alternative is to average over standard deviation estimates, each obtained by a different bootstrap sample. Yet this version was found to be inferior in our evaluation.
[7] The unnormalized variants were significantly inferior to the normalized ones.

	AP		TREC4		TREC5		Robust		WT10g		GOV2	
Method	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	.596	.428	.456	.380	.490	.258	.477	.328	.380	.240	.407	.305
QF	.575	.385	.632	.570	.413	.310	.483	.371	.436	.343	.515	.383
WIG	.526	.380	.533	.502	.347	.252	.411	.358	.434	.364	.535	.387
σ_{100}	.381	.306	.335	.302	.270	.261	.468	.378	.475	.334	.421	.305
$\sigma_{\max}$	.381	.309	.411	.331	.333	.227	.344	.347	.405	.296	.402	.301
$\sigma_{50\%}$	.341	.308	.278	.254	.202	.191	.395	.331	.376	.333	.383	.289
$n(\sigma_{50\%})$	.510	.396	.301	.281	.231	.240	.461	.367	.436	.350	.411	.302
σ_k	.475	.370	.487	.402	.450	.329	.501	.386	.352	.301	.402	.286
NQC	.554	.361	.624	.562	.483	.318	.575	.406	.486	.354	.432	.304
UEF[NQC]	.613	.394	.639	.569	.520	.332	.615	.418	.518	.361	.455	.327
SMV	.631	.398	.524	.499	.459	.268	.586	.432	.292	.206	.418	.304
Bagging[org]	.678	.462	.629	.567	.557	.369	.593	.412	.519	.330	.530	.365
Bagging[rank]	.682	.457	.632	.571	.564	.375	.606	.430	.529	.348	.537	.369
RSD[uni]	.703^{ob}	.469^{ob}	.633	.574	.590$^{ob}_{s}$	.405^{ob}	.622$^{ob}_{s}$	.438$^{ob}_{s}$	.553$^{ob}_{s}$	.366^{b}	.568$^{ob}_{s}$	.371$_{s}$
RSD[sim]	.694^{ob}	.466^{o}	.634	.571	.570^{ob}	.395^{ob}	.606	.421	.524^{o}	.362^{o}	.553	.363
RSD[wig]	**.710**$^{ob}_{us}$	**.473**^{ob}	**.651**^{b}	**.581**^{b}	**.618**$^{ob}_{us}$	**.421**$^{ob}_{us}$	**.649**$^{ob}_{us}$	**.441**$^{ob}_{us}$	**.561**$^{ob}_{us}$	**.387**$^{ob}_{us}$	**.576**$^{ob}_{us}$	**.403**$^{ob}_{us}$

Table 2: Comparison between the RSD variants and the alternative baseline methods. Bold values mark the best performing method per usecase. The superscripts *o* and *b* denote a statistically significant better performance of a given RSD variant compared to all other alternative baselines that use only the original (single) result list for prediction and the two Bagging variants, respectively. The subscripts *u* and *s* further denote statistically significant better performance of either the RSD[uni] or RSD[wig] variants compared to the RSD[uni] and RSD[sim] variants, respectively. All significance notations are further reported using a Bonferroni correction for $p < 0.05$.

3.4.2 Impact of RSD's bootstrap sampling approach. We next evaluate the impact of RSD's bootstrap sampling approach by comparing its RSD[uni] variant with the two Bagging variants. We first note that, these two Bagging variants, in most cases, also outperform the prediction quality of the other alternative baseline methods which only make use of the original result list. Among the two Bagging variants, Bagging[rank] was (slightly) better.

Similar to Bagging, the RSD[uni] variant treats every sample evenly. Hence, we can evaluate the impact of our proposed bootstrap sampling approach that is employed in RSD[uni] by directly comparing it with that employed by the Bagging[rank] variant. To recall, compared to Bagging, RSD does not allow document repetitions and further utilizes a round-robin sampling scheme. Overall, RSD[uni] outperformed the prediction quality of Bagging (an average improvement of 3.6%(±0.8%) and 2.6%(±1.5%) in P-ρ and K-τ, respectively). This is yet another empirical testimony that, our new proposed bootstrap sampling approach, which is inspired by user search behavior, is better tailored for QPP.

3.4.3 Comparison of RSD variants. Among the three RSD variants that we evaluated, RSD[wig] had the best prediction quality, with an average improvement of 2.6%(±0.6%) and 2.9%(±1.4%) in P-ρ and K-τ, respectively, over the next best RSD variant (which was RSD[uni] in most cases). Notable is the significant difference between RSD[sim] and RSD[wig]. This empirical result implies that, a better reference lists prediction combination strategy would probably be to use another QPP method (in our case a variant of WIG was utilized) rather than to use inter-list similarity as was originally proposed in [12].

REFERENCES

[1] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996.

[2] David Carmel and Oren Kurland. Query performance prediction for ir. In *Proceedings of SIGIR '12.*

[3] David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. What makes a query difficult? In *Proceedings of SIGIR '06.*

[4] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of WSDM '08.*

[5] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *Proceedings of SIGIR '02.*

[6] Ronan Cummins, Joemon Jose, and Colm O'Riordan. Improved query performance prediction using standard deviation. In *Proceedings of SIGIR '11.*

[7] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of SIGIR '01.*

[8] Joaquín Pérez-Iglesias and Lourdes Araujo. Standard deviation as a query hardness estimator. In *Proceedings of SPIRE'10.*

[9] Haggai Roitman. An enhanced approach to query performance prediction using reference lists. In *Proceedings of SIGIR '17.*

[10] Haggai Roitman, Oren Sar-Shalom, Shai Erera, and Bar Weiner. Enhanced mean retrieval score estimation for query performance prediction. In *Proceedings of ICTIR '17.*

[11] Anna Shtok, Oren Kurland, and David Carmel. Using statistical decision theory and relevance models for query-performance prediction. In *Proceedings of SIGIR '10.*

[12] Anna Shtok, Oren Kurland, and David Carmel. Query performance prediction using reference lists. *ACM Trans. Inf. Syst.*, 34(4):19:1–19:34, June 2016.

[13] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. Predicting query performance by query-drift estimation. *ACM Trans. Inf. Syst.*, 30(2):11:1–11:35, May 2012.

[14] Mor Sondak, Anna Shtok, and Oren Kurland. Estimating query representativeness for query-performance prediction. In *Proceedings of SIGIR '13.*

[15] Yongquan Tao and Shengli Wu. Query performance prediction by considering score magnitude and variance together. In *Proceedings of CIKM '14.*

[16] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4):20:1–20:38, November 2010.

[17] C. X. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. *In Proceedings of SIGIR '01.*

[18] Yun Zhou and W. Bruce Croft. Query performance prediction in web search environments. In *Proceedings of SIGIR '07.*