Exploring Reductions for Long Web Queries

Niranjan Balasubramanian*
University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003
niranjan@cs.umass.edu

Giridhar Kumaran and Vitor R. Carvalho
Microsoft Corporation
One Microsoft Way, Redmond, WA
{giridhar,vitor}@microsoft.com

ABSTRACT

Long queries form a difficult, but increasingly important segment for web search engines. Query reduction, a technique for dropping unnecessary query terms from long queries, improves performance of ad-hoc retrieval on TREC collections. Also, it has great potential for improving long web queries (upto 25% improvement in NDCG@5). However, query reduction on the web is hampered by the lack of accurate query performance predictors and the constraints imposed by search engine architectures and ranking algorithms.

In this paper, we present query reduction techniques for long web queries that leverage effective and efficient query performance predictors. We propose three learning formulations that combine these predictors to perform automatic query reduction. These formulations enable trading off average improvements for the number of queries impacted, and enable easy integration into the search engine's architecture for rank-time query reduction. Experiments on a large collection of long queries issued to a commercial search engine show that the proposed techniques significantly outperform baselines, with more than 12% improvement in NDCG@5 in the impacted set of queries. Extension to the formulations such as result interleaving further improves results. We find that the proposed techniques deliver consistent retrieval gains where it matters most: poorly performing long web queries.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Theory

Keywords

Query reformulation, Learning to Rank, Combining Searches

*This work was done when the author was an intern at Microsoft Corporation.

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

1. INTRODUCTION

Long queries form a sizable fraction of the queries that are submitted to web search engines. Queries of length five words or more have increased at a year over year rate of 10%, while single word queries dropped 3% [1]. Unfortunately, web search engines perform poorly on longer queries when compared to shorter ones [4].

Several past works have focused on improving long query performance [3, 15, 14, 16, 13, 7] (see Section 2). They can be broadly classified into query re-weighting and query reduction approaches. In query re-weighting, the original query terms are assigned different weights before the query is submitted to the search engine. In query reduction, instead of the original query, a reduced version of it (obtained by removing one or more terms from the original query) is selected and submitted to the search engine.

Both approaches are motivated by the observation that long queries usually contain superfluous terms, which if down-weighted or completely removed, result in improved performance. For example, consider the query *easter egg hunts in northeast columbus parks and recreation centers*, which performs moderately well on most popular web search engines. If we remove (or down-weight in some fashion) the terms *and recreation centers*, we can observe a perceptible improvement in the quality of results.

While query term re-weighting and reduction techniques have shown significant improvements in performance on TREC[1] data [15, 13], their utility in the web environment is not well understood. Further, query reduction/re-weighting on the web poses unique challenges due to the architecture of web search engines, and the extremely low latencies tolerated. Query re-weighting requires the use of a ranking algorithm that permits the online assignment of weights to query terms — a difficult thing to implement given that most web retrieval algorithms use a learning to rank framework that relies on boolean match as well as several additional query-dependent and query-independent features. Query reduction requires analyzing a potentially exponential number of reduced versions of the original query, and relies heavily on query quality prediction measures like Clarity that are expensive to compute on the web.

In this paper, we focus on query reduction, and develop techniques that are especially suited for the web. We utilize an efficient, rank-time query quality prediction technique [2], and consider only reduced versions of the original query that are obtained by dropping a single term at a time. As we will show in Section 3.2, dropping just a single (and correct!)

[1]http://trec.nist.gov

term from the original long query can result in a 26% improvement in NDCG@5. After formally defining the query reduction problem in Section 3.1, we propose three different formulations of the problem in Section 3.3. We demonstrate their utility (Section 5) through experiments on a set of approximately 6400 long queries sampled from the query logs of a major web search engine. Additionally, we develop simple thresholding and interleaving extensions to these formulations that enable balancing the trade-off between the queries impacted and the improvements attained.

The main contributions of this paper are (1) a rank-time query reduction technique that is well suited for incorporation in a web search engine and works reliably in improving *hard* long queries (2) formal treatment of the query reduction technique that allows control over when and how to do query reduction, and (3) the first large scale evaluation of query reduction for web queries. Analysis of the experimental results show that query reduction achieves consistent gains whenever there is high potential for improvement (Section 6). More importantly, we find that query reduction mostly benefits poorly performing queries. In other words, it provides benefits where they are most needed.

2. RELATED WORK

Retrieval effectiveness for long queries is often lower than retrieval effectiveness for shorter keyword queries. Kumaran et al. [12] showed that shorter queries extracted from longer user generated queries are more effective for ad-hoc retrieval. Using click positions as surrogates for effectivenessm Bendersky and Croft [4] show that long query effectiveness is lower than short keyword queries for web queries. Several works have focused on improving the retrieval performance of long queries [3], [15], [14], [16],[13], and [7]. They can be broadly categorized as automatic term weighting and query reduction approaches.

Weighting - Bendersky and Croft [3] use a key concepts classifier to identify important query terms in long queries, and to assign weights to query terms resulting in improved retrieval effectiveness. Also, Bendersky et al. [5] successfully extend this term weighting approach to concepts and build a weighted dependence model that performs well for web queries. Lease et al. [15] use Regression Rank, a regression framework to directly learn weights on query terms using query dependent secondary features. An extension to this approach achieves significant improvements over Markov Random Field dependency models on TREC collections [14].

However, as mentioned earlier it is difficult to directly incorporate term weights in existing web search engine ranking algorithms. In contrast, our goal is develop an approach that can be seamlessly incorporated into existing web search engines architectures without requiring modifications to the underlying search algorithms.

Reduction - Kumaran and Carvalho [13] develop an automatic method for reducing long TREC description queries. Using content-based query quality predictors such as Clarity and Mutual Information Gain, they convert the query reduction task into a problem of ranking (reduced) queries based on their predicted effectiveness. Their results on TREC Robust 2004 show the viability of automatic query reduction. In this work, we investigate the extension of this approach to web search engines.

Lee et al. [16] use statistical and linguistic features of query terms to greedily select query terms from the original long query to achieve the effect of query reduction. Experiments on NTCIR collections demonstrate that this approach, and its extension which considers pairs of terms [17] improves long queries' effectiveness.

Chen et al. [7] use personal query history to select short queries related to the original long query, cluster the short queries based on similarity of their contexts, and select representatives from each cluster as a substitution for the original long query. Unfortunately, this approach requires rank-time processing of texts of retrieved documents, extracting noun phrases and other features from sentences that contain the query words. These heavy rank-time computations make this approach infeasible for use in web search engines.

In summary, the term weighting, query reduction, and substitution approaches show good potential for improving long query effectiveness. However, term weighting approaches cannot be readily incorporated into web search engines, while the features used for the various query reduction approaches are not suitable for efficiency reasons. Furthermore, there has been no large scale experiments on actual long web queries that demonstrate the utility of automatic query reduction. In this paper, we investigate the utility of query reduction for long web queries, and develop formulations that are well-suited for Web search engines.

3. QUERY REDUCTION

Retrieval effectiveness is typically lower for longer queries than for shorter keyword queries, partly because users often utilize extraneous terms in long queries [12]. Query reduction - the technique of automatically identifying and removing extraneous terms from long queries - has proved to be an effective technique for improving performance on long queries [13]. In this section, we present a formal description the query reduction problem.

3.1 Formal Description

Let $f : \mathcal{P} \times D \rightarrow \mathbb{R}$ denote a ranking function that scores documents (D) with respect to a query P, represented as a *set* of query terms. Also, let $T_f(P)$ denote a target measure of the effectiveness of the ranking produced by f for the query P.

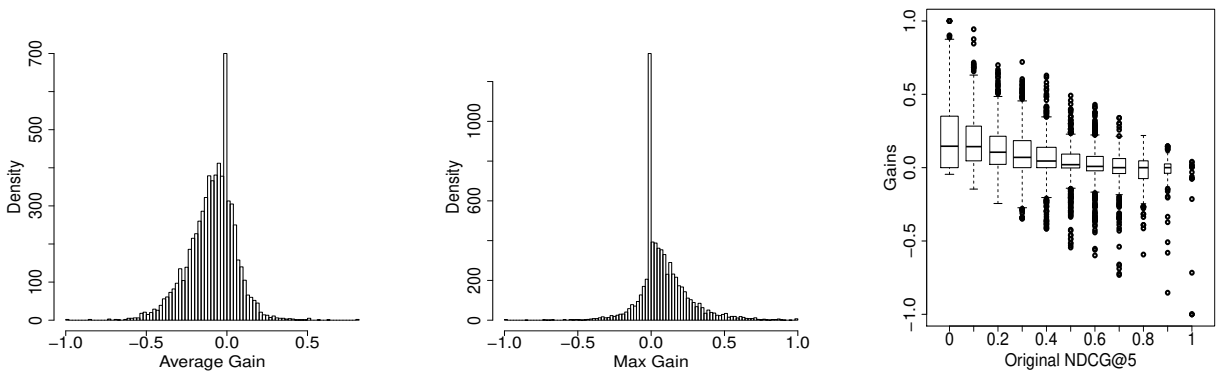
Given an arbitrary query $Q = \{q_1, \cdots, q_n\}$, we use $\mathcal{P}^Q$ to denote the power set of Q, i.e., the set containing all subsets of terms from query Q (including the original query Q). Then, the query reduction problem is to find a reduced version P^* that achieves the highest value for the target measure as shown in Equation 1. Note that this problem statement allows the original query Q to be selected as well.

$$P^* = \arg\max_{P \in \mathcal{P}^Q} T_f(P) \tag{1}$$

Obviously, the target measures cannot be completely specified for inferences over all possible queries, and hence we need to estimate $T_f(P)$. The query reduction task is then expressed as:

$$P^* = \arg\max_{P \in \mathcal{P}^Q} \widehat{T_f}(P) \tag{2}$$

Query performance predictors, such as *Clarity* [8] or *Query Scope* [9], can be used to obtain estimates of the target measure, $\widehat{T_f}$ in order to select a reduced version P^* of the original query Q.



(a) Average Gains (b) Max Gains (c) Original versus Gains

Figure 1: Distribution of Potential Gains in NDCG@5.

3.2 Approximation

Efficiency is a key challenge for query reduction. Because the number of possible reduced queries in $\mathcal{P}^Q$ is exponentially large, enumerating and evaluating all possibilities is not feasible. This challenge is even more important for web search engines, where response times are in the order of milliseconds.

To address this issue, we propose a simpler version of the query reduction problem. In particular, instead of considering all possible reduced versions, we only consider those that differ from the original query Q by only one term. That is, instead of using the entire power-set $\mathcal{P}^Q$, we use a restricted version $\mathcal{P}_1^Q = \{P|P \in \mathcal{P}^Q \wedge |P| \geq |Q| - 1\}$. Thus, if the original query had n query words, we only need to consider the n reduced versions and original query Q.

Despite the obvious limitation of ignoring a large number of potentially better reduced versions, this simple approach can yield dramatic performance improvements. On a large collection of more than 6400 Web queries (see Section 4 for details), we find that an oracle that chooses between an original long query and its reduced versions achieves more than 10 points gain in NDCG@5.

However, in order to achieve this gain, we need to reliably identify reduced versions whose performances are better than that of the corresponding original queries. To illustrate the potential impact of this technique, we analyze the distribution of maximum and average gains for reduced versions using Figures 1(a), (b), and (c). Figures 1(a) and (b) show the distribution of gains when compared to the original query. Figure 1(c) shows distribution of gains (as box plots) for original queries with different NDCG@5 values.

On average, the reduced versions' effectiveness are worse compared to the original query's effectiveness, as shown by the negative gains dominant in Figure 1(a). Also, the maximum gains, the gains that can be achieved if we always identify the best reduced version, are mostly positive as shown in Figure 1(b). However, for some queries the maximum gains are negative i.e., choosing any reduced version will result in decreased performance. Finally, Figure 1(c) shows that if the original query has poor performance, then it is more likely for some reduced version to be better than the original query. Conversely we are unlikely to find reduced versions of well-performing queries that provide substantial performance gains.

Based on these observations, we develop learning techniques that can reliably improve the performance of *hard* long web queries through query reduction.

3.3 Learning Formulations

We use three formulations for choosing between the original query and its reduced versions: 1) *Independent* performance prediction, 2) *Difference* prediction, and 3) *Ranking* queries. All three formulations use the same performance predictors to generate features but differ in their target learning functions. For the remainder of this paper, we assume that the same ranking algorithm, f, is used to retrieve results for both the original query and its reduced versions and hence drop it from our notations.

Let $\mathbb{Q}$ be the set of training queries and let $T(\mathbb{Q})$ be the effectiveness of their retrieved results.

3.3.1 Independent Prediction

Given an original long query and its reduced versions, we predict the performance of each query independently. Then, we select the query that has the highest predicted performance. Thus, the query selection problem is transformed into a query performance prediction task: Given a query, and the retrieved results, the task is to predict the effectiveness of the retrieved results.

Formally, given a set of functions $h : \mathcal{P}^Q \rightarrow \mathbb{R}$, we learn a non-linear regressor h^* that minimizes the mean squared error as given by:

$$h^* = \arg\min_h \sqrt{\sum_{\forall Q \in \mathbb{Q}, P \in P_1^Q} (h(P) - T(P))^2}$$

For a given test query Q_t, we select the query P^* with the largest predicted performance, i.e.:

$$P^* = \arg\max_{P \in \mathcal{P}_1^{Q_t}} h^*(P) \quad (3)$$

3.3.2 Difference Prediction

While the *Independent* formulation is relatively simple, it does not encode the relationship between the original query and its reduced versions. Furthermore, based on the observations from Figure 1(c), it may be more important to predict the difference in performance between the original query and its reduced versions, than to accurately predict the effectiveness of the individual queries.

In the *Difference* formulation, we predict the difference in performance between each reduced version and its original query, and then select the query that has the highest positive difference. If there is no reduced version with a predicted positive difference, then we choose the original query.

Let $D(Q,P) = T(P) - T(Q)$, denote the target measure difference between a reduced version P and its original query Q. Given a set of functions $h_d : Q \times Q \rightarrow \mathbb{R}$, we learn a least-squared-errors regressor h_d^* given by:

$$h_d^* = \arg\min_{h_d} \sqrt{\sum_{Q \in \mathbb{Q}} \sum_{P \in \mathcal{P}_1^Q \wedge P \neq Q} \left(h_d(Q,P) - D(Q,P)\right)^2}$$

For a given test query, Q_t, we choose a reduced representation P^* as:

$$P^* = \arg\max_{P \in \mathcal{P}_1^{Q_t}} h^*(Q,P) \quad (4)$$

3.3.3 Ranking Queries

In this formulation, the goal is to rank the original query and its reduced versions in order to select the top ranking query. The ranking model is learned by training on pairwise preferences between the queries.

For each reduced version $P \in \mathcal{P}_1^Q$, P is preferred over Q if $T(P) \geq T(Q)$. The pairwise preferences induce a partial ordering and the query at the top of the ordering is selected. This formulation fully encodes dependencies between the original query, and all the reduced versions. Kumaran and Carvalho [13] successfully use this learning to rank approach to select *only* amongst reduced versions of the query on TREC collections. In this work, we also include the original query in addition to the reduced versions.

Let Φ denote the error function for incorrect pairwise ordering defined as follows:

$$\Phi_h(Q,P) = \begin{cases} 1 & \text{if } \text{sign}\left(h(P) - h(Q)\right) \neq \text{sign}\left(T(P) - T(Q)\right) \\ 0 & \text{otherwise} \end{cases}$$

We want to learn a function h_r^* from the set of ranking functions $h_r : Q \rightarrow \mathbb{R}$ such that it minimizes the overall ranking errors, i.e.,:

$$h_r^* = \arg\min_{h_r} \sum_{Q \in \mathbb{Q}} \sum_{P \in \mathcal{P}_1^Q} \Phi_h(Q,P)$$

For a given test query, Q_t, we choose a reduced representation P^* as:

$$P^* = \arg\max_{P \in \mathcal{P}_1^{Q_t}} h_r^*(P) \quad (5)$$

3.4 Thresholding

As an extension to these formulations, we also learn a threshold on the assigned scores in order to control the number of queries for which reduced versions are selected. In *Independent*, a reduced version is selected if and only if, there exists a reduced version whose predicted performance is greater than that of the original query by a specified threshold. For *Difference*, the positive difference has to exceed a threshold in order to choose a reduced version. Finally, for *Ranking*, the predicted performance of the top-ranking reduced version must exceed the original query's predicted performance by the specified threshold.

For all three formulations, we also learn the best threshold values by selecting the values that achieve the best improvements over the training set of queries.

3.5 Performance Prediction

Accurate prediction of performance of the original query and reduced versions is critical to the success of the query reduction formulations we have described in this Section. We leverage a rank-time technique for query performance prediction [2]. The key idea behind this technique is to utilize retrieval scores, and the features that a ranker uses for ranking documents to estimate the quality of the results. Table 1 lists the two broad types of features used for ranking documents as well as predicting query quality – those that characterize the prior probability of query effectiveness, and those that characterize the quality of the retrieved results. In *Independent* the features are used as is, while in *Difference* and *Ranking* the difference in feature values between the original and the reduced versions are used.

Query Features : We use several query-specific features to provide a richer representation of the query. Lexical features flagging the presence of URL, stop words, and numbers as well as location features that denote the presence of town, city, or state names are part of the feature set. Additionally, we use query length as a feature. Query length has a strong negative correlation with performance i.e., retrieval effectiveness degrades as query length increases.

Query-Document Features: The retrieval scores assigned by the ranking algorithm are indicative of the relevance of individual documents, and aggregates of these scores are useful predictors of the relevance of the retrieved results. Web search engines often combine multiple rank-time features to score documents. These rank-time features provide different types of evidence for the relevance of the documents. Some useful features include query-independent features similar to page-rank, query-log based features such as variations of click-through counts, and term match-based features such as BM25F.

Using these query and document-related features, we train a regressor to directly predict the performance (NDCG@5) of queries.

4. EXPERIMENTAL SETUP

We conduct experiments to evaluate query reduction and the utility of the different learning formulations.

We use LambdaRank [6] as our web ranking algorithm. LambdaRank has been shown to be an effective learning to rank technique for the web and can handle a large number of features. We target NDCG@5 [10] as the metric to maximize. We use the top 5 results to create the query-document features, and select the top 100 features that are most correlated with NDCG@5 in the training data. We evaluate the query reduction approach and the utility of the different formulations on a large collection of Web queries. To create the query reduction collection, we first obtain a frequency-weighted random sample[2] of more than 6400 long queries issued to a major Web search engine. For each query in this collection, we also create reduced versions by dropping a single word each time. We use the LambdaRank retrieval algorithm to rank documents for both the original

[2]The frequency-weighted sampling ensures the chosen queries are a representative mix of the types of long queries.

Table 1: Features used for performance prediction. Pos. indicates that the value for the particular row is computed for each top-k document and used as an independent feature. Agg. indicates that statistical aggregates (mean, max, standard deviation, variance and coefficient of dispersion) of the values for the top-k documents are also used as independent features. Additionally, we perform a min-max normalization to rescale all features between $[0, 1]$.

Type	Feature Name	Description	Variants
Query	URL	Binary: Does query contain an URL?	-
	Stop-words	# of stop-words in query	-
	Location	Does query contain a town, city name or a state name ?	-
	Query Length	# of words in query	-
Query-Document	LR	LambdaRank score of top-k documents	Pos. and Agg.
	BM25	Okapi-BM25 score of top-k documents	Pos. and Agg.
	Click-based	Click-through counts and other variants	Agg.
	Static Scores	Page-rank like scores of the top-k documents	Agg.

long query and all its reduced versions. Then, we obtain relevance judgments for both the results for the original query and the reduced versions from a large pool of independent annotators. The reduced versions results are also judged with respect to the original long query, and not with respect to the reduced one. We refer to this collection as *Long Web* henceforth.

Learning Algorithms. For *Independent* and *Difference* formulation, the goal of learning is to find real-valued functions that predicts the target metric, and the differences in the target metric, respectively. For both formulations, we use non-linear regression with the Random Forests [18] algorithm[3], to predict performance of queries and performance differences respectively. For the *Ranking* formulation, we use RankSVM [11] [4] to learn pair-wise preferences. For all three problem formulations we perform five-fold cross validation for training and evaluating the learning models.

5. RESULTS

For each formulation, we report results for two types of experiments. In Query Replacement experiments, if a reduced version is selected, it is used to replace the original query. In Results Interleaving if a reduced version is selected, the results of the selected reduced version and the original query are interleaved. The interleave order depended on the sign of the difference between their predicted NDCG@5.

5.1 Query Replacement

Table 2 shows the performance of the different problem formulations (top half) along with the thresholding extension (bottom half) for the Query Replacement task. We compare the formulations using two measures: 1) Overall NDCG@5, which is the macro-averaged NDCG@5 over all queries, and 2) Subset NDCG gain, the average improvements on the subset for which reduced versions were chosen.

With no thresholding, *Difference* achieves the best overall gain. *Ranking*'s overall gain is lower but the subset gain is substantially higher. While, *Difference* and *Ranking* both achieve small but significant overall gains, *Independent* is actually worse than the original. We believe that this performance difference is due to two reasons. First, *Difference* and *Ranking* encode the relationship between the original query and its reduced versions, whereas *Independent* does not capture such relationships. Second, the *Independent* formulation appears to solve a harder learning problem. The regression in *Independent* attempts to minimize mean-squared errors of the predicted and actual NDCG@5 values. The accuracy of the predicted values matter only in terms of the ordering they induce, the difference between the absolute values are not important. In fact, we find that the root-mean squared errors for *Independent*'s regression was 10% worse compared to that of *Difference*'s regression, even though both regressions are learned using the same learning algorithm, the same set of base features, and same training data.

With thresholding however, *Independent* improves dramatically and its overall performance is comparable to *Difference*. On the other hand, *Difference* and *Ranking* do not achieve any additional improvements to the overall measures due to thresholding. *Independent* benefits from thresholding as it selects fewer reduced versions which are more likely to yield improvements. Despite the similar average performance (over the entire set of queries), the three formulations provide different types of improvements. First, *Independent* and *Ranking* select reduced versions for less than 10% of the queries, whereas *Difference* selects reduced versions for more than 27% of the queries. The number of selected reduced versions depends upon the accuracy of the predicted scores, and the thresholds learnt during training.

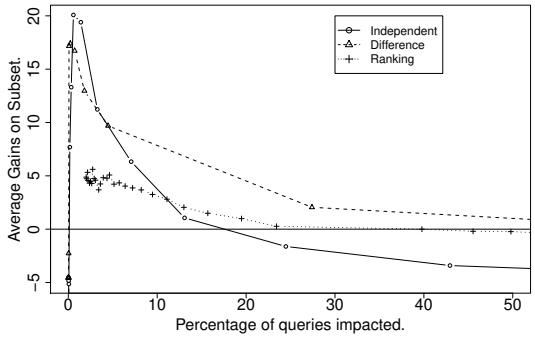


Figure 2: Number of queries affected versus improvements in subset gains.

To better understand the relationship between the number of queries affected versus the average improvement in performance, we explore the behavior of the different formulations at various thresholds. In general, we expect increasing thresholds to cause fewer reduced versions to be chosen (lower recall), but to also increase the likelihood of

[3]We used the *randomForest* package available from R with default parameters

[4]We used the SVMLight implementation with default parameters and a linear kernel

Table 2: Effectiveness of Query Replacement. The baseline i.e., using the original query, has an overall NDCG@5 of 38.19. Bold face indicates the highest value, and * indicates statistically significant improvement over the baseline ($p < 0.05$ on a two-tailed paired t-test).

	Overall NDCG@5	Affected Queries	Improved Queries	Hurt Queries	Subset NDCG Gain
No Thresholding					
Independent	35.18	4567 (70%)	1583	2346	- 4.26 (-12%)
Difference	38.63*	1761 (27%)	513	427	+ 1.61 (+4.2%)
Ranking	38.50*	612 (9%)	245	212	+ 4.64(+12.1%)
Thresholding					
Independent	**38.64***	457 (7%)	219	149	+ 6.33 (+16.5%)
Difference	38.63*	1761 (27%)	513	427	+ 1.61 (+4.2%)
Ranking	38.50*	612 (9%)	245	212	+ 4.64(+12.1%)

improving over the original query (higher precision). Figure 2 shows this precision-recall trade-off in terms of the gains achieved on the subset of affected queries against the percentage of queries affected. As expected for all three formulations, the average performance on the subset is drastically high for a small percentage of queries, and performance decreases as more queries are affected. *Independent* and *Difference* achieve more than 10 points absolute improvement over the original queries on a subset of 5% of the queries. Compared to *Independent* and *Difference*, *Ranking* achieves smaller gains but retains its performance over a large fraction of queries. This suggests that *Ranking* is able to choose reduced versions effectively for more number of queries but it may not always choose the best reduced version.

Feature Importance. The contribution of the different feature groups for *Difference* are shown in Table 3. Most of the gains come from the *query-document* features. Adding the *query features* provides small improvements. As expected features that characterize the result set directly are more useful than the *query-features*. Also, adding estimates of original query's effectiveness further improves performance. This is because reduced versions are more likely to improve poorly performing original queries and adding estimates helps *Difference* to encode this relationship.

Table 3: Feature importance for *Difference*: *Orig* — using original query, with no query replacement. *QueryDoc* — using query-document features alone. *+Query* — adding query features. +Estim — adding estimate of the performance of the original query.

	Orig	QueryDoc	+ Query	+ Estim
Overal Gain	38.19	38.52	38.63	38.73
Subset Gain	0	1.30	1.61	2.05

5.2 Results Interleaving

To reduce the risk involved in choosing between queries, we conduct interleaving experiments. In all three formulations, if a reduced version is selected, we interleave its results with the results of the original query. Furthermore, we decide the order of interleaving based on the predicted performance. If the original query's predicted performance was higher than that of the reduced version then interleaving begins with the original query, and vice-versa otherwise[5]

Table 4 shows the gains achieved by the interleaving results. *Difference* achieves the best overall gains, whereas

[5]Because interleaving combines results from the original query and the top-ranked reduced version, it can yield gains even in cases where the top-ranked reduced version's predicted performance is lower than that of the original query.

Table 4: Results Interleaving at the best thresholds for the three formulations. For the NDCG Gain row, bold face indicates the highest value, and * indicates statistically significant improvement over original NDCG@5 ($p < 0.05$ on a paired t-test).

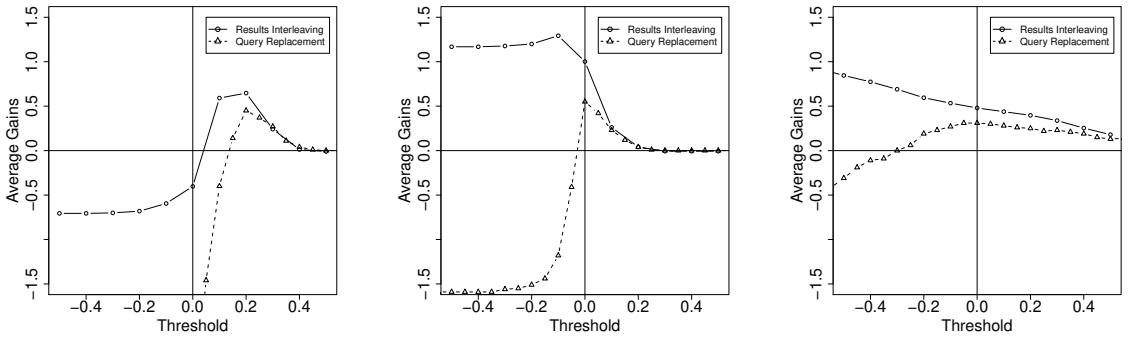
	Indep.	Diff.	Rank.
Overall Gain	0.7*	**1.3***	0.97*
Subset Gain	9.89	1.3	1.19
Best Threshold	0.2	-0.2	-0.8
Affected Queries	457	6435	5258
Improved Queries	228 (4%)	2052 (31%)	1620 (25%)
Hurt Queries	139 (2%)	2063 (31%)	1612 (25%)

Independent achieves the best subset gains. *Difference* and *Ranking* both have a positive impact on a large number of queries, 31% and 25% respectively, whereas *Independent* provides positive gains for only 4%. We hypothesize that one of the reasons that *Difference* and *Ranking* achieve higher performance compared to *Independent* is because *Difference* and *Ranking* achieve better ranking of reduced versions compared to *Independent*, thus allowing more queries to benefit from interleaving.

Figures 3(a), (b), and (c), show Results Interleaving and Query Replacement performance at different thresholds for all formulations. For all formulations, Results Interleaving performs better compared to Query Replacement at all thresholds. Since interleaving ensures that at least some of the original query's results are mixed in with the chosen reduced versions results, we reduce the risk of hurting performance in case of erroneous choices. However, interleaving also benefits from the fusion of results from the original long query, and the reduced version in the case of good choices. Although not shown here, an analysis of the gains obtained by Query Replacement, and Results Interleaving for *Independent* formulation shows that there are fewer queries with large positive gains for Results Interleaving compared to Query Replacement, but there are also fewer negative gains. For example, nearly 10% of positive gains in Query Replacement are above 25 points, whereas only 5% of positive gains in Results Interleaving are above 25 points. On the other hand, nearly 20% of the negative gains in Query Replacement are below 20 points, whereas less than 5% of the negative gains in Results Interleaving are below 20 points.

6. ANALYSIS

We further analyze the results to better understand the distribution of gains and the nature of the improvements.



(a) Independent: NDCG@5 (b) Difference: NDCG@5 (c) Ranking: NDCG@5

Figure 3: Interleaving Results.

Query reduction results in dramatic gains on some subset of queries but also incurs losses on some queries. For Query Replacement using *Independent* formulation nearly 60% of this impacted queries were improved. Further, for 20% of these queries, the gain was more than 50 points in absolute NDCG. This indicates the dramatic improvements attainable through query reduction. For the remaining 40%, a large proportion of their losses are small, which explains the overall improvements. We observe similar trends for *Difference* and *Ranking*.

Potential versus Achieved Gains. All three formulations provide improvements when there is a large potential for improvement. Figure 4 shows the distribution of the gains achieved by *Independent* in relation to the best gains that an oracle can achieve. In most cases when the potential for improvement is large, *Independent* formulation achieves larger improvements. Also, When the potential is greater than 0.8, *Independent* always results in some positive improvement. We believe that the large gains achieved by the formulations are primarily due to two factors.

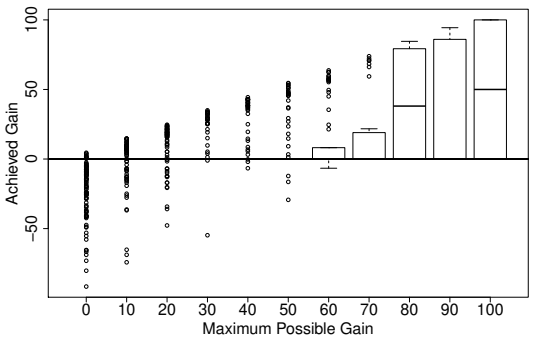
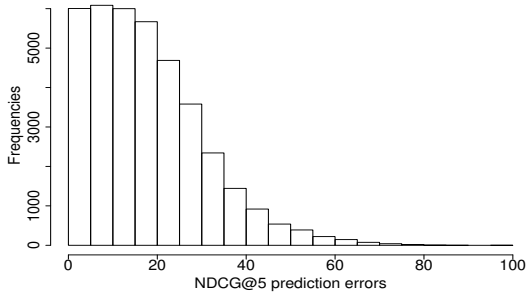


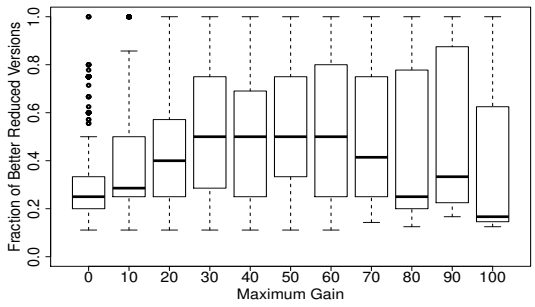
Figure 4: Oracle versus Achieved Gains: Boxplot showing the distribution of gains achieved by Query Replacement using *Independent* in relation to the best possible gains that can be achieved by an oracle.

First, the formulations are able to detect large differences in NDCG@5 more reliably. The distribution of absolute prediction errors for the regression used by *Independent* is shown in Figure 5(a). The histogram shows the frequencies of absolute difference between the predicted NDCG@5 and the acutal NDCG@5 values for all queries (including reduced versions). Most prediction errors are smaller, and very few errors are larger than 50 points (less than 3%). This suggests that smaller differences in NDCG@5 are harder to capture given the range of prediction errors, while larger differences can be captured more reliably.

Second, for queries with large gains, the problem of choosing a reduced version becomes easier. Figure 5(b) shows the distribution of number of reduced versions that are better than the original query. The number of better reduced versions correlates with the maximum achievable gain (shown in the x-axis). For queries with high potential for improvement (i.e., queries with high maximum gain), the number of better reduced versions is higher, which makes the problem of finding better reduced versions easier.



(a) NDCG@5 Prediction Errors: Frequency histogram of difference between predicted and actual NDCG@5 values.

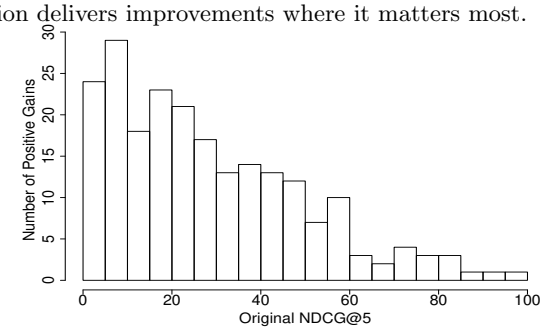


(b) Boxplot showing distribution of reduced versions that are better than the original query versus potential gains.

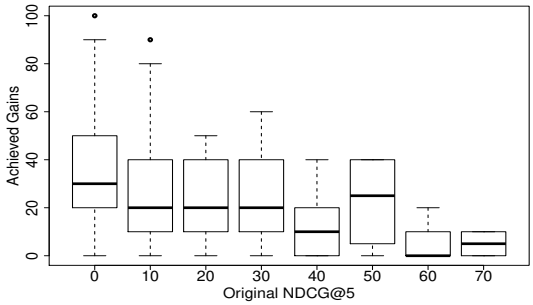
Figure 5: Prediction accuracy and difficulty of choosing reduced versions.

Improving Poorly Performing Queries. As illustrated in Figure 1, poorly performing original queries often have large potential for improvements. Since all formulations deliver improvements when there is large potential, most gains are achieved for poorly performing original queries. Figure 6(a) shows the histograms for the number of queries that achieve positive gains against the effectiveness

of the original query. Clearly, most of the gains are achieved for queries whose original NDCG@5 low. Nearly 75% of the queries that benefit from *Independent* are queries with NDCG@5 ≤ 40. Further, the magnitude of the gains for the poorly performing queries are higher than for well performing queries as shown in Figure 6(b). Thus, unlike traditional techniques such as pseudo-relevance feedback, query reduction delivers improvements where it matters most.



(a) Number of queries at each effectiveness level which achieved positive gains.



(b) Boxplot showing magnitude of achieved gains for original queries of different effectiveness levels.

Figure 6: Performance of *Independent* for original queries of different effectiveness levels

7. CONCLUSIONS

As the average length of queries users submit to search engines increases, long query retrieval becomes an increasingly important problem. In this paper we have addressed some of the key challenges in adapting query reduction for long web queries — a particularly difficult task due to the inherent constraints imposed by modern search engine architectures and operational requirements.

We presented three learning formulations that naturally provide different trade-offs in terms of number of queries affected versus the overall average gains achieved by query reduction. Such flexibility is valuable when designing large scale systems, where memory and processing limitations may significantly impact when and whether a query should be altered.

We also provided the first comprehensive evaluation on a large collection of real long web queries. Our experiments showed that directly predicting performance differences generally outperforms independent performance predictions. Also, performance of the proposed formulations can be improved even further by interleaving results from the original and reduced versions. A careful analysis of the results clearly showed that, unlike traditional techniques such as pseudo-relevance feedback, our query reduction techniques achieved most NDCG gains on difficult long queries.

The naïve approximation to the full scale (exponential) query reduction problem substantially improves efficiency (exponential to linear), while still providing significant effectiveness gains. However, even evaluating a linear number of additional queries can be burdensome for search engines. Also, despite improved average performance, we find that there is high variance in performance. As part of future work, we aim to further improve efficiency and reduce variance using a two-staged approach that first predicts the effectiveness of the original long query to decide when to evaluate the reduced versions.

8. ACKNOWLEDGMENT

This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed here are the authors' and do not necessarily reflect those of the sponsor. The authors also thank the anonymous reviewers for their helpful comments and suggestions.

9. REFERENCES

[1] Searches getting longer: A weblog by alan long, hitwise intelligence. `http://weblogs.hitwise.com/alan-long/2009/11/searches_getting_longer.html`.

[2] N. Balasubramanian, G. Kumaran, and V. Carvalho. Predicting query performance on the web. In *SIGIR 2010*.

[3] M. Bendersky and W. Croft. Discovering key concepts in verbose queries. In *SIGIR*, pages 491–498, 2008.

[4] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log. In *WSCD*, pages 8–14, 2009.

[5] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *WSDM '10*, pages 31–40, 2010.

[6] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. *NIPS*, 19:193, 2007.

[7] Y. Chen and Y.-Q. Zhang. A query substitution - search result refinement approach for long query web searches. In *WI-IAT*, pages 245–251, 2009.

[8] C. Hauff, V. Murdock, and R. Baeza-Yates. Improved query difficulty prediction for the web. In *CIKM*, pages 439–448, 2008.

[9] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *SPIRE*, pages 43–54, 2004.

[10] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142, 2002.

[12] G. Kumaran and J. Allan. A case for shorter queries, and helping users create them. In *HLT/NAACL*, pages 220–227, 2007.

[13] G. Kumaran and V. Carvalho. Reducing long queries using query quality predictors. In *SIGIR*, pages 564–571, 2009.

[14] M. Lease. An improved markov random field model for supporting verbose queries. In *SIGIR*, pages 476–483, 2009.

[15] M. Lease, J. Allan, and W. B. Croft. Regression rank: Learning to meet the opportunity of descriptive queries. In *ECIR*, pages 90–101, 2009.

[16] C. Lee, Y. Lin, R. Chen, and P. Cheng. Selecting Effective Terms for Query Formulation. In *AIRS 2009*, pages 168–180, 2009.

[17] C.-J. Lee, R.-C. Chen, S.-H. Kao, and P.-J. Cheng. A term dependency-based approach for query terms ranking. In *CIKM '09*, pages 1267–1276, 2009.

[18] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.