Reducing Long Queries Using Query Quality Predictors

Giridhar Kumaran
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA
giridhar@microsoft.com

Vitor R. Carvalho
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA
vitor@microsoft.com

ABSTRACT

Long queries frequently contain many extraneous terms that hinder retrieval of relevant documents. We present techniques to *reduce* long queries to more effective shorter ones that lack those extraneous terms. Our work is motivated by the observation that perfectly reducing long TREC description queries can lead to an average improvement of 30% in mean average precision. Our approach involves transforming the reduction problem into a problem of learning to rank all sub-sets of the original query (sub-queries) based on their predicted quality, and select the top sub-query. We use various measures of query quality described in the literature as features to represent sub-queries, and train a classifier. Replacing the original long query with the top-ranked sub-query chosen by the ranking classifier results in a statistically significant average improvement of 8% on our test sets. Analysis of the results shows that query reduction is well-suited for moderately-performing long queries, and a small set of query quality predictors are well-suited for the task of ranking sub-queries.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation

General Terms

Algorithms, Experimentation, Performance

Keywords

Long Queries, Verbose Queries, Query Quality, Query Reduction

1. INTRODUCTION

While most queries presented to search engines vary between one to three terms in length, a gradual increase in the average length of queries has been observed[1]. These longer queries are typically used to convey more sophisticated information needs. Unfortunately, the performance of most commercial and academic search engines deteriorates while handling longer queries. For example, a query such as *ideas for breakfast menu for a morning staff meeting*, while conveying the true information need of the user, is better handled by most search engines when posed as *breakfast meeting menu ideas*. This task of *reducing* the original query to a shorter one in the course of a search session is usually left to the user.

In this paper we present a way to automatically reduce long queries to shorter, more effective ones. Long queries lend themselves to such reduction as they invariably contain extraneous terms (*for*, *morning*, and *staff* in our example) that serve more to confuse the search engine than support it in its task. In Section 2, we analyze and quantify the potential improvement in mean average precision (MAP, Section 4) that can obtained by selecting ideal terms (or rejecting unnecessary terms) from long TREC *description* queries.

This potential for improvement motivated past work like Bendersky and Croft [1] and Lease et al. [18] on automatic techniques for query reduction. The former's approach involved learning to identify key concepts in long queries using a variety of features while the latter focused on a regression-based approach to re-weight all the terms in long queries. An interactive technique that involved completely dropping unnecessary terms from long queries was successfully demonstrated by Kumaran and Allan [17]. However, the interaction burdened the user with additional cognitive and physical effort.

Our technique for automatic query reduction involves analyzing all the subsets of terms from the original query (sub-queries), and identifying the most promising sub-query to replace the original long query. The technique involves representing each sub-query by a set of query quality indicators, and then learning effective ranking functions based on this representation.

To find indicators or features to represent (the quality of) each sub-query, we draw on the large body of previous work on query quality prediction. These include predicting the quality (or performance) of queries using either pre-retrieval indicators like Query Scope [10], or post-retrieval indicators like Query Clarity [7]. The ability to predict query performance finds use in applications as diverse as resource selection in federated search [26], determining whether to invoke

SIGIR'09, July 19–23, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-483-6/09/07 ...$10.00.

[1] http://blogs.zdnet.com/micro-markets/index.php?p=27

user interaction [16], and learning when to advertise [2]. We used query-quality predictors as features to describe each sub-query. In Section 3 we describe the entire set of query quality predictors that we used.

Our approach to ranking sub-queries is reminiscent of work related to learning to rank documents [13, 4, 22]. Learning to rank documents involves representing documents using features computed from properties associated with them, and the query. A classifier is then trained to rank relevant documents higher. In the paper we apply this framework to the problem of learning to rank sub-queries, with the specific goal of surfacing the best sub-query to replace the original long query. The particular algorithm we used is outlined in Section 4.2.

We performed query reduction experiments on several combinations of TREC collections (Section 4), and observed significant improvements in performance (Section 5). These observations validated the utility of query reduction. The analysis of the results, which is presented in Section 6, revealed that query reduction is most useful for long queries that originally exhibited moderate performance as measured by MAP. Further, from among the thirty one features that we used, we observed that a small subset of well-known query quality predictors like query clarity provided most of the predictive power in the models we learned for different collections. All this points to query reduction as a promising avenue for future work, with emphasis on developing better query quality predictors and effective strategies and algorithms designed for ranking queries.

2. MOTIVATION

In this section, we provide an illustrative example showing the utility of query reduction, and demonstrate the improvement in performance that can be realized through the technique. We further show how the problem of query reduction can be transformed into a ranking problem.

2.1 Utility of Reduction

TREC topics consist of a *title*, *description*, and *narrative*, of progressively increasing length. While the title is usually between one and four terms in length, the description is longer, ranging from three to thirty terms in length. We use the TREC description queries as surrogates for long queries in our experiments.

Table 1 provides insight into the utility of query reduction for the description portion of TREC topic 333 from the Robust 2004 data collection. The table contains the actual description query in the header, along with sample sub-queries and their associated performance metrics in the first two columns (Sub-query and Performance Metrics). We can observe that some sub-queries achieve significantly better performance compared to the original query. These queries are the candidate targets for query reduction.

Table 2 shows the summary upper-bound performance that can be achieved for a set of 200 TREC *description* queries from the Robust 2004 track used for training (Section 4). "Baseline" refers to a query-likelihood retrieval model [20] run using the Indri search engine [21] using the original long query. "Oracle" refers to the situation when the best sub-query was selected to replace the original long query. This gives us an upper bound on the performance that can be realized through query reduction for this set of queries. It is this statistically significant improvement in performance through query reduction that we target in our work.

System	P@5	P@10	NDCG@15	MAP
Baseline	0.452	0.398	0.373	0.241
Oracle	**0.588** +30%	**0.508** +27.6%	**0.479** +28.4%	**0.297** +23.2%

Table 2: The utility of query reduction for 200 training queries from the TREC 2004 Robust collection. A value in bold face implies statistically significant improvement over the baseline. Statistical significance was measured using the Wilcoxon matched-pairs signed-ranks test, with α set to 0.05.

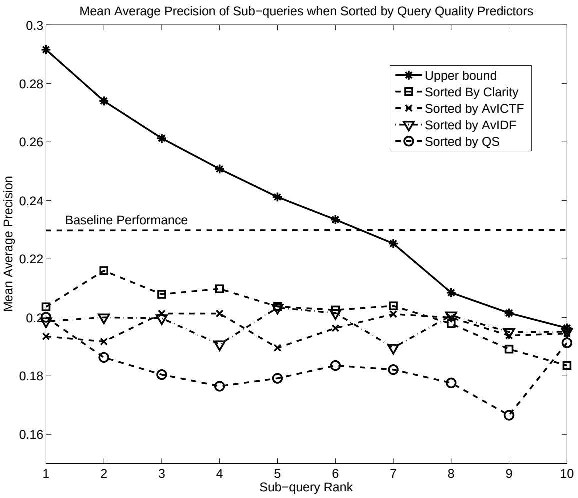


Figure 1: Mean average precision at various ranks when sub-queries are ranked using query quality predictors. The sub-queries are from 172 Robust 2004 Track long description queries.

2.2 Sub-query selection as ranking

We now focus our attention on the third column of Table 1 (Query Quality Predictors). This column contains the values of some popular query quality predictors for each of the sub-queries: Query Clarity [7], average inverse collection term frequency (AvICTF) [10], average inverse document frequency (AvIDF), and query scope (QS) [10]. Also included in the final row are the correlation coefficient values of each of these query predictors with the AP of sub-queries. The values imply a weak–positive to positive correlation between the predictors and AP. This means that if we were to use these query quality predictors to rank the sub-queries, then there is a reasonable chance that we can identify a better-performing sub-query.

In Figure 1 we illustrate the effect of ranking sub-queries of 172 long training[2] queries from the Robust 2004 track using individual query quality predictors. The figure shows the mean average precision (MAP) at ranks one to ten when

[2] We selected training queries that had ten or more sub-queries to generate this graph. Our sub-query pruning procedure (Section 4.3) resulted in some queries having less then ten sub-queries.

TREC Topic 333	Determine the reasons why bacteria seems to be winning the war against antibiotics and rendering antibiotics now less effective in treating diseases than they were in the past.							
Sub-query	Performance Metrics				Query Quality Predictors			
	P@5	P@10	NDCG@15	AP	Clarity	AvICTF	AvIDF	QS
effect bacteria antibiotic	0.800	0.900	0.880	**0.395**	4.950	14.615	15.161	1.858
bacteria antibiotic render	0.800	0.900	0.883	**0.383**	5.017	16.182	16.075	4.520
bacteria determine antibiotic	0.800	0.800	0.777	**0.373**	5.382	14.925	15.336	2.433
bacteria determine antibiotic treat	0.800	0.700	0.727	**0.348**	5.308	14.062	14.747	2.038
bacteria determine antibiotic disease	0.800	0.800	0.773	**0.342**	4.771	14.443	15.073	2.316
bacteria antibiotic disease war	0.800	0.800	0.785	**0.337**	3.692	14.415	15.098	2.397
effect bacteria antibiotic disease	0.600	0.800	0.737	**0.336**	4.233	14.211	14.941	1.803
effect bacteria antibiotic disease render	0.800	0.800	0.744	**0.335**	4.027	14.303	14.903	1.772
effect bacteria antibiotic treat	0.800	0.900	0.647	**0.334**	4.837	13.830	14.615	1.646
bacteria antibiotic past disease	0.800	0.800	0.731	**0.334**	4.508	14.390	14.997	2.019
bacteria antibiotic treat	0.800	0.700	0.646	**0.332**	4.725	15.116	15.484	2.904
bacteria antibiotic disease	0.800	0.800	0.744	**0.332**	4.338	15.624	15.918	4.167
bacteria antibiotic disease render	0.800	0.800	0.792	**0.330**	4.605	15.386	15.627	3.737
bacteria antibiotic treat render	0.800	0.800	0.725	**0.328**	4.424	15.005	15.301	2.776
reason bacteria antibiotic disease	0.800	0.800	0.643	0.322	3.846	14.369	15.000	2.038
...	...	...	...	...	...	...	...	...
bacteria antibiotic disease treat render	0.600	0.700	0.662	0.298	4.456	14.604	15.097	2.641
effect bacteria antibiotic disease treat render	0.600	0.700	0.527	0.293	4.110	13.832	14.583	1.589
effect bacteria antibiotic disease treat	0.600	0.600	0.512	0.286	4.143	13.664	14.549	1.613
antibiotic disease treat	0.200	0.300	0.327	0.208	3.457	13.897	14.723	2.757
bacteria disease treat	0.200	0.100	0.162	0.071	4.109	13.710	14.609	2.754
	Correlation coefficient				*0.507*	*0.473*	*0.446*	*0.020*

Table 1: Sample sub-queries for the description portion of TREC Robust 2004 Topic 333. The baseline query, which is the original query (provided in the header row) after stemming and stop word removal, had an average precision (AP) of 0.327. The sub-queries have been sorted in the order of decreasing AP. We can observe that there are fourteen sub-queries that have a higher AP than the original long query.

the sub-queries are ranked using each of the query quality predictors, as well as the upper bound ranking for the 172 queries. We notice that none of the query quality predictors is able to achieve the upper bound performance, or even the baseline performance got by using the long queries without any modifications. We hypothesize that by using a combination of the query quality predictors we can train a ranker that beats the baseline performance.

This is the main idea of the paper: using query quality predictors as features to learn a ranking function to target the upper-bound ranking of sub-queries. Once the ranking is completed, we can use the top sub-query in place of the corresponding long query. In the next section, we will describe the set of features we considered for each sub-query, and the machine learning algorithms we used to learn a ranking function to bias better performing sub-queries higher up the ranked list.

3. QUERY QUALITY PREDICTORS

For each of the $O(2^n)$ sub-queries that a query of length n can have, we calculated the following query quality predictors to serve as descriptive features. While some of these features such as mutual information (MI) are stand-alone, i.e. they are calculated for the sub-query as a whole, other features such as Average IDF (AvIDF) are derived from term level statistics. Some of these features are pre-retrieval, i.e. they are derived directly from query and corpus statistics. Others like Query Clarity are post-retrieval, i.e. they involve performing an initial retrieval and hence are more *expensive* to compute. We now describe the set of query quality predictors we used, and include references to their sources.

3.1 Mutual Information (MI)

This feature was adopted from the work by Kumaran and Allan [15] that was based on the observation by van Rijsbergen [23]. We represented each of the $O(2^n)$ sub-queries as a graph constructed with the constituent terms as vertices, and the mutual information [5] between the terms as edge weights. The mutual information was calculated using Equation 1 [5]. The maximum spanning tree [6] was then identified on each graph, and its average weight was used as a predictor of the quality of the corresponding sub-query.

$$I(x,y) = log\frac{\frac{n(x,y)}{T}}{\frac{n(x)}{T}\frac{n(y)}{T}} \qquad (1)$$

where $n(x,y)$ is the number of times terms x and y occurred within a term window of 100 terms across the corpus, $n(x)$ and $n(y)$ are the frequencies of x and y in the collection and T is the number of term occurrences in the collection.

3.2 Sub-query Length (SQLen)

Drawing on work by He and Ounis [10] and the observation by Kumaran and Allan [17] that the best sub-queries have lengths between two and six, we included SQLen as a query quality predictor. SQLen for a sub-query is defined as the number of terms in it.

3.3 Query Clarity (QC)

Developed by Cronen-Townsend et al. [7] this post-retrieval predictor is the Kullback-Leibler divergence of the query model from the collection model. The query model is estimated from the top-ranked documents retrieved by the original query. QC is computed as

$$QC = \sum_{w \in Q} P(w|Q) \times log_2 \frac{P(w|Q)}{P_C(w)}$$

where $P(w|Q)$ is the probability of the occurrence of the word w in the query model, and $P_C(w)$ is the probability of the occurrence of w in the collection.

3.4 Simplified Clarity Score (SCS)

To avoid the expensive computation of query clarity, He and Ounis [10] proposed simplified clarity score as a comparable pre-retrieval performance predictor. It is calculated as

$$SCS = \sum_{w \in Q} P_{ml}(w|Q) \times log_2 \frac{P_{ml}(w|Q)}{P_C(w)}$$

where $P_{ml}(w|Q)$ is the probability of the occurrence of the word w in the query.

3.5 IDF-based features

We calculated the IDF of each query term w as

$$IDF_w = \frac{log_2 \frac{N+0.5}{N_w}}{log_2(N+1)}$$

where N_w is the document frequency of w and N is the number of documents in the collection.

For each sub-query we calculated the (a) sum (b) standard deviation [10] (c) maximum/minimum [10] (d) maximum (e) arithmetic mean (f) geometric mean (g) harmonic mean and (h) coefficient of variation of the IDFs of constituent terms. These values served as additional query quality predictors for each sub-query.

3.6 Query Scope (QS)

Query scope [10, 19] is a measure of the size of the retrieved document set relative to the size of the collection. We can expect that high values of query scope are predictive of poor-quality queries as they retrieve far too many documents.

$$QS = -log \frac{n_Q}{N}$$

where n_Q is the number of documents containing at least one query term.

3.7 Similarity Collection/Query-based features (SCQ)

Proposed by Zhao et al. [28], this query quality predictor is based on the hypothesis that queries that have higher similarity to the collection as a whole will be of higher quality. For each term w in the query

$$SCQ_w = (1 + \ln(\frac{n(w)}{N})) \times \ln(1 + \frac{N}{N_w})$$

Based on the SCQ values of each term, we calculated aggregate values similar to those for IDF (Section 3.5) as subquery quality predictors.

3.8 Inverse Collection Term Frequency-based features (ICTF)

Inverse collection term frequency of a term w is defined as

$$ICTF_w = log_2 \frac{n(w)}{T}$$

Using the ICTF values, we calculated aggregate statistics similar to those for IDF (Section 3.5).

3.9 Similarity Original Query (SOQ)

Guided by the notion that the reduced query should still reflect the original query's information need, we calculated the cosine similarity between the TF-IDF vectors representing each sub-query and the original long query. We hypothesized that a sub-query that was not radically different from the original query was preferable to one that had drifted away completely from the original query's intent.

In summary, we represented each sub-query with a set of thirty one features. In the following sections we will describe the experimental setup as well as the results of learning to rank sub-queries using these features.

4. EXPERIMENTAL SETUP

4.1 Collections

We used the indexing and retrieval capabilities of version 2.6 of the Indri search engine, developed as part of the Lemur[3] project. Our retrieval model was the query-likelihood variant of statistical language modeling [20]. We used Dirichlet smoothing [27] with μ set to 1000. For experimentation, we used three collections[4]. The first, Robust 2004, was the 2004 Robust track collection that has 250[5] queries and contains around half a million documents from the Financial Times, the Federal Register, the LA Times, and FBIS. The second collection, TREC123, was created by combining existing TREC collections. It contained documents from TREC disks 1 and 2, and the 150 TREC topics 51 to 200 as queries. We took advantage of the fact that some TREC collections used the same sets of documents, and created this new collection with a larger number of queries. This enabled the creation of good training/test splits required for training classifiers. Each set of queries was broken down into an 80%/20% split of train and test queries. The Robust 2004 collection is known to contain difficult queries, and thus provided a challenging data set to test the utility of query reduction for hard queries. TREC123 offered a collection of moderate difficulty. To further test the ability to learn a ranking across collections, we combined the training (200 + 120) and test (46 + 30) queries from Robust 2004 and TREC123 to create a new collection called Robust 2004 + TREC123.

All collections were stemmed using the Krovetz stemmer [14] provided as part of Indri. We used an extended set of 418 stop words, also referred to as the INQUERY stop word list [3]. To identify named entities in the queries, we

[3] `http://www.lemurproject.org`

[4] We refer to a set of documents and associated queries as a collection.

[5] We had to drop four queries that either did not have any relevant documents in the collection, or were too long to be handled by our system.

used the Stanford Named-Entity Recognizer [8], which can identify Person, Location, and Organization entities.

As performance measures, we report precision at five documents (P@5), precision at ten documents (P@10), normalized discounted cumulative gain at 15 documents (NDCG@15, as defined in [24]), and mean average precision (MAP). P@5 and P@10 refer to the fraction of relevant documents in the top five and ten documents retrieved respectively. NDCG@15 is a measure similar to precision that includes rank-based discounting. This means that systems that return relevant documents higher up a ranked list will receive higher scores compared to those that return them lower. Average precision (AP) is a single value obtained by averaging the precision values at each new relevant document observed. MAP is the arithmetic mean of the average precisions of a set of queries.

4.2 Ranking Algorithm and Training

Given as input a set of predictors for each sub-query, our goal was to combine these inputs to produce an effective ranking function. To accomplish this we used RankSVM [13], a learning-to-rank algorithm based on the same framework as the well-established Support Vector Machines (SVM) classification algorithm.

For each original query Q_i of length n, the set of all possible $O(2^n)$ sub-queries $SQ_i = \{sq_{i1}, sq_{i2}, ..., sq_{i2^n}\}$ was generated. Each sub-query was represented by its AP value y_{ij} and associated vector of k query quality predictors $sq_{ij} = [x_{ij1}, x_{ij2}, ..., x_{ijk}]$. RankSVM works by utilizing a pairwise preference ranking framework in which instead of taking each sub-query in isolation, all possible sub-query pairs (with different AP values) are used as instances in the learning process.

We used the RankSVM implementation available in the SVM^{Light} [12] package. Both Linear and RBF kernels were considered in our experiments. The regularization parameter C (trade-off between training error and margin) as well as the *gamma* parameter of the RBF kernel were selected from a search within the discrete set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ over the validation set[6]. For the selection of parameter C, the default SVMLight value was also considered.

Although the differences were not substantial, experiments with the best RBF kernel parameters performed slightly better than the best linear kernel parameters for the majority of the validation experiments. Unless otherwise noted, all results henceforth were obtained using an RBF kernel, with *gamma* set to 0.001.

4.3 Pruning Sub-query Candidates

An exponential number ($O(2^n)$) of sub-queries can be obtained from a query of length n. Further, training the ranking classifier involved generation of pairwise preference constraints from this exponential number. This 2^n followed by n^2 combinatorial explosion resulted in an extremely large training set that was very expensive to train using SVM^{Light}. This necessitated the pruning of candidate sub-queries. We used the following strategies to reduce the number of candidate sub-queries.

- Select only sub-queries with length between three and six terms. Kumaran and Allan [17] showed that the best sub-queries for most long queries had lengths in the range mentioned.
- Select only the top twenty five sub-queries ranked by the MI feature (Section 3.1) for consideration. Our initial experiments suggested that this was one of the most predictive features, and hence we decided to use it in the first pass to identify strong candidates for reduction.
- Select only sub-queries that contained named entities. Again, following the results from Kumaran and Allan [17] we considered only sub-queries that contained at least one of the named entities from the original query, if it contained any.

These simple pruning strategies resulted in a more manageable set of candidate sub-queries. We acknowledge that such pruning could hurt performance, but as we will show in the next section, we still obtained significant improvements in performance on our test queries.

5. RESULTS

Table 3 contains the results of ranking the sub-queries of the test queries for each of the collections we considered. We can observe that our ranking technique selects query reductions that result in a statistically significant improvement over the baseline for all collections. Also included in the table are the upper bound values for the performance metrics, i.e. the performance when a hypothetical ranker achieves perfect ranking of the test sub-queries. This provides an indication of the scope for further improvement.

We notice that the rankers we have trained achieve 15 – 30% of the net gain in MAP possible for the three collections. The percentage improvement is greater for TREC123 than for Robust 2004, possibly indicative of the moderate difficulty of the TREC123 queries. The improvement due to query reduction for Robust 2004 + TREC123 is encouraging as it implies that the ranking procedure is robust enough to handle training and test instances containing feature values computed from different collections.

6. ANALYSIS

Query Performance: Query reduction resulted in statistically significant improvements in MAP for all the test collections. To understand the effect of query reduction on long queries, we now analyze the results introduced above. To make the task of analysis easier, we define three types of long queries. The first, with an baseline MAP between 0 and 0.1, are considered poorly-performing queries. Queries with a baseline MAP between 0.1 and 0.4 are labeled moderately-performing queries. Finally, queries with a baseline MAP greater than 0.4 are referred to as well-performing queries. The distribution of queries of different types, as well as the observations we make, are specific to the collections we have experimented with. Additional collections need to be analyzed to check if the same trends carry over.

Figure 2 is a set of three scatter plots depicting the utility of query reduction on all three collections. To provide clarity, incomplete grid lines have been drawn to isolate the areas corresponding to the queries of different types. The line $y = x$ is also included to convey whether a particular

[6] 20% of the queries in the training set were randomly selected and used to create a validation set.

Collection	Query	P@5	P@10	NDCG@15	MAP
TREC123	Original Long Query	0.527	0.487	0.493	0.219
	Top-ranked Sub-query	0.520	0.507	0.500	**0.241** (+10.0%, p = 0.027)
30 test queries	Oracle	0.647	0.620	0.614	0.282 (+28.7%)
Robust 2004	Original Long Query	0.465	0.407	0.409	0.249
	Top-ranked Sub-query	0.491	0.422	0.430	**0.266** (+6.8%, p = 0.018)
46 test queries	Oracle	0.587	0.474	0.514	0.318 (+27.7%)
Robust 2004 + TREC123	Original Long Query	0.487	0.438	0.443	0.239
	Top-ranked Sub-query	0.495	0.441	0.445	**0.253** (+5.8%, p = 0.007)
76 test queries	Oracle	0.603	0.529	0.551	0.305 (+27.6%)

Table 3: The results of ranking sub-queries and selecting the top-ranked sub-query to replace the original long query. Values in bold indicate that the performance improvement was statistically significant when measured using the Wilcoxon matched-pairs signed-ranks test, with α set to 0.05. We can observe that the improvements in P@5, P@10, and NDCG@15 follow the trends for MAP.

Rank	Robust 2004	TREC123	Robust 2004 + TREC123
1	Clarity	Clarity	Clarity
2	IDF_{max}/IDF_{min}	Mutual Information	Mutual Information
3	Total ICTF	Coeff. Of Variation(SCQ)	Max ICTF
4	Total IDF	Total ICTF	Total IDF
5	Mutual Information	Max ICTF	Total ICTF

Table 4: Scale-normalized ranking of the most important features in different collections.

query was improved by query reduction or not. A point above the line corresponds to a query that was improved by query reduction, while a point below the line refers to a query that was hurt by query reduction.

Figure 2(a) is a scatter plot for the 30 test queries in TREC123. The bottom left box contains queries that are poorly-performing, the middle one contains queries that are moderately-performing, and the rightmost one is occupied by high-performing queries. Query reduction for poorly-performing queries almost never results in significant improvements in performance. An approximately equal number of queries appear to be hurt and improved. In the box for high-performing queries, where there are fewer queries, we estimate that query reduction leads to much larger gains or losses in performance. The bulk of the overall improvement in performance that we see appears to come from moderately-performing queries. The improvements for this category are neither as pronounced as the high-performing queries nor are they as minuscule as for the poorly-performing ones. We notice similar trends in all three query sets.

Another interesting point to note is that the queries appear to be "boxed-in", i.e. very rarely do we see a query that is drastically hurt or drastically improved to the extent that is ends up outside the boxes. The size of the boxes increases progressively from poorly-performing to high-performing queries. This means that the effect of query reduction is more pronounced as the quality of the original query improves.

Robust 2004 + TREC123 (Figure 2(c)) contains a mix of relatively easy TREC123 queries and hard Robust 2004 queries. We observe that the trends observed in Figures 2(a) and 2(b) carry over to this collection as well, i.e. the ranker trained on Robust 2004 + TREC123 performs better on moderately performing TREC123 queries (marked with symbol 'o') than on moderately performing Robust 2004 queries (marked with symbol 'x'). Thus, the quality of the original query plays a very important role in the impact of query reduction. Increasing the amount of training data, as in the case of Robust 2004 + TREC123, doesn't seem to result in a ranker more capable of improving poor and moderately performing long queries.

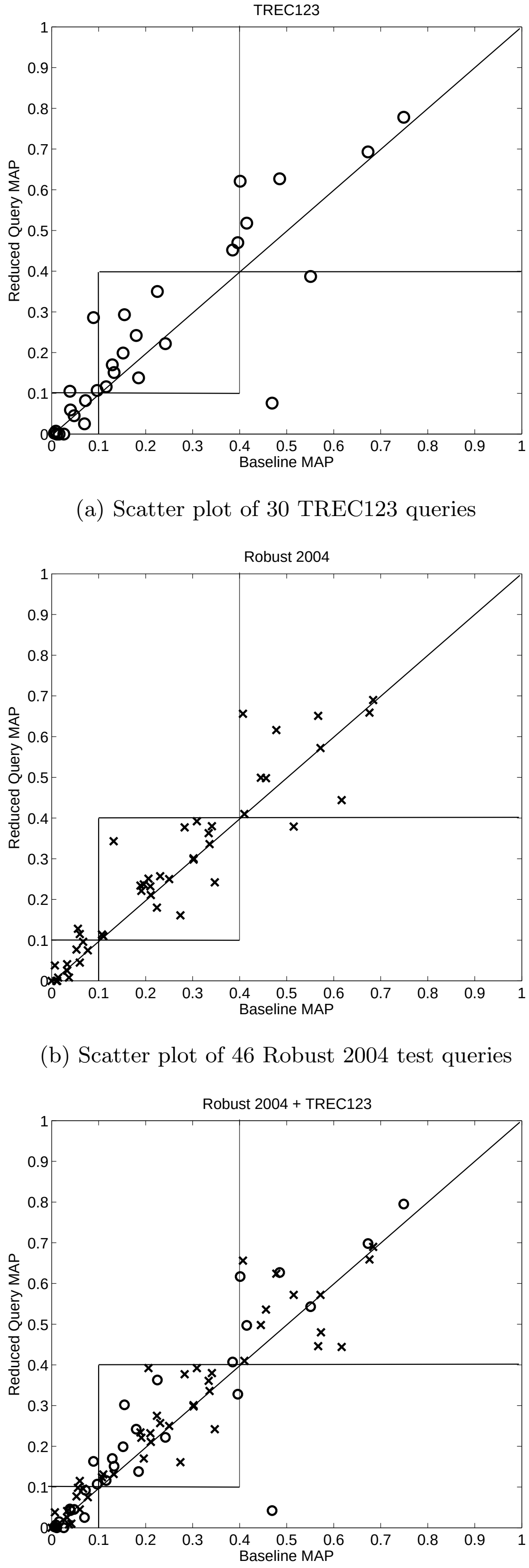
Top Features: In Table 4 we present the list of five top-ranked features for each collection. To identify these features we used RankSVM with linear kernels since the relative feature weights are not available for models with RBF kernels[7]. The table shows that the set of top features is quite consistent across collections though their relative importance depends on the collection. Clarity and features based on term-coherence (MI), IDF and ICTF are clearly the most important features.

7. RELATED WORK

Past work by Kumaran and Allan [15] set the stage for interactive versions of query reduction. Using a single feature namely mutual information (MI) they selected a set of ten top-ranked sub-queries and presented them to the user to choose from. By analyzing supplemental information in the form of snippets of text from top-ranked documents corresponding to sub-queries, users were able to select good sub-queries. However, such selection required additional physical and cognitive effort from users, motivating the need for automatic query reduction techniques.

Bendersky and Croft [1] approached the problem of query reduction as a problem of finding key concepts in long queries for preferential weighting. They used a number of query and corpus-dependent and corpus-independent features to learn to identify the key terms in long queries. Our work differs from theirs as we did not assume that finding key concepts will capitalize on the full potential for query

[7] In all collections, the linear kernel ranker performed slightly worse than the RBF one, but the difference was not statistically significant.



(a) Scatter plot of 30 TREC123 queries

(b) Scatter plot of 46 Robust 2004 test queries

(c) Scatter plot of 76 Robust 2004 + TREC123 queries

Figure 2: Scatter plots of baseline long queries MAP versus MAP of corresponding reduced queries.

reduction. With the exception of named entities, we treat all terms as equally useful following the observation by Kumaran and Allan [15] that the best sub-queries sometimes contained terms that could be considered ordinary, but help with retrieval in more complex ways than imagined.

More recently, Lease et al. [18] proposed a regression framework to re-weight the terms in long queries. They introduced a set of secondary features that correlated with the term weights, and then applied different types of regression techniques to learn appropriate feature-based ranking functions. In contrast with our approach, Lease et al. did not try to reduce long query terms directly, but instead choose to re-weight them.

Numerous efforts have been made towards improving techniques for predicting query quality. Cronen-Townsend et al. [7] developed the Query Clarity measure, which was the top-ranking feature in all our models, to serve as a predictive measure for tracking MAP. He and Ounis [10] explored a number of pre-retrieval features to determine query effectiveness. Zhao et al. [28] explored pre-retrieval predictors that were based on the similarity between a query and the document collection as well as the variability in query term distribution across documents. Features built on the query-collection similarity-based predictors ranked high in the list of important features identified by our rankers.

Hauff et al. [9] conducted a survey of 22 pre-retrieval query quality predictors. They concluded that there wasn't one single predictor that performed best on all collections, and the utility of different predictors was related to the collection in question. In contrast, our experiments used both pre and post retrieval query quality predictors, and results revealed that clarity was consistently the best feature for all test collections.

Unlike MI, our query-term coherence measure, He et al. experimented with the use of measures related to the coherence of documents in the ranked list [11]. While they reported those measures as being correlated to MAP, we choose not to use them because of the complexity involved in calculating the measures' values.

8. CONCLUSIONS

We have presented a new way of approaching the problem of query reduction, an effective technique that is quite hard to realize. By casting the query reduction problem as a sub-query ranking problem we have been able to draw on work in the areas of query quality prediction and learning to rank. We have shown statistically significant improvements on all our test collections, validating the utility of query reduction. Our analysis of the results revealed some interesting properties of long queries such as the dependence of the utility of query reduction on the quality of the original long query. By analyzing the top features in the learned ranking models we have identified a set of features well-suited for ranking sub-queries. Our choice of query quality predictors is by no means exhaustive. However, our choice of predictors was deliberate - we avoided complex features like document and ranked list perturbation [29, 25] would have been too expensive to compute for the exponential number of sub-queries. By showing significant improvement in performance using easily computed features, we have paved the way for easy adoption of query reduction.

The quality of a ranker is intimately connected to the quality of the features used to represent training and test instances. As future work we plan to work on developing and incorporating more effective query quality predictors. We plan to extend this work on TREC collections to queries submited to web search engines. We also plan to optimize the ranking of sub-queries to target other performance measures like P@5, P@10, and NDCG.

Acknowledgments

We wish to thank the anonymous reviewers of this paper for their very helpful comments.

9. REFERENCES

[1] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 491–498, New York, NY, USA, 2008. ACM.

[2] A. Broder, M. Ciaramita, M. Fontoura, E. Gabrilovich, V. Josifovski, D. Metzler, V. Murdock, and V. Plachouras. To swing or not to swing: learning when (not) to advertise. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1003–1012, New York, NY, USA, 2008. ACM.

[3] J. Callan, W. B. Croft, and J. Broglio. Trec and tipster experiments with inquery. *Information Processing and Management*, 31(3):327–343, 1994.

[4] V. R. Carvalho, J. Elsas, W. W. Cohen, and J. G. Carbonell. A meta-learning approach for robust rank learning. In *SIGIR-2008 LR4IR (Workshop on Learning to Rank for Information Retrieval)*, Singapore, 2008.

[5] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *27th ACL Proceedings*, pages 76–83, 1989.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, II Edition*. MIT Press, 2001.

[7] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, New York, NY, USA, 2002. ACM.

[8] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[9] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1419–1420, New York, NY, USA, 2008. ACM.

[10] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *The Eleventh Symposium on String Processing and Information Retrieval*, 2004.

[11] J. He, M. Larson, and M. de Rijke. Using coherence-based measures to predict query difficulty. In *30th European Conference on Information Retrieval (ECIR 2008)*, pages 689–694. Springer, Springer, April 2008.

[12] T. Joachims. Making large-scale support vector machine learning practical. pages 169–184, 1999.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[14] R. Krovetz. Viewing morphology as an inference process. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–202, New York, NY, USA, 1993. ACM.

[15] G. Kumaran and J. Allan. A case for shorter queries, and helping users create them. In *HLT-EMNLP Conference Proceedings*, pages 220–227, Rochester, NY, 2007.

[16] G. Kumaran and J. Allan. Selective user interaction. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 923–926, New York, NY, USA, 2007. ACM.

[17] G. Kumaran and J. Allan. Effective and efficient user interaction for long queries. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, New York, NY, USA, 2008. ACM.

[18] M. Lease, J. Allan, and B. Croft. Regression rank: Learning to meet the opportunity of descriptive queries. In *Proceedings of the 31st European Conference on Information Retrieval (ECIR)*, 2009. To appear.

[19] V. Plachouras, F. Cacheda, I. Ounis, and C. J. V. Rijsbergen. Rijsbergen. university of glasgow at the web track: Dynamic application of hyperlink analysis using the query scope. In *In Proceedings of the Twelth Text REtrieval Conference (TREC 2003*, page 248, 2003.

[20] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.

[21] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *International Conference on Intelligence Analysis*, 2005.

[22] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *CIKM '06: ACM international Conference on Information and Knowledge Management*, pages 585–593, New York, NY, USA, 2006.

[23] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2 edition, 1979.

[24] S. Vassilvitskii and E. Brill. Using web-graph distance for relevance feedback in web search. In *29th ACM SIGIR Proceedings*, pages 147–153, 2006.

[25] V. Vinay, I. J. Cox, N. Milic-Frayling, and K. Wood. On ranking the effectiveness of searches. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 398–404, New York, NY, USA, 2006. ACM.

[26] 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 *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 512–519, New York, NY, USA, 2005. ACM.

[27] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[28] Y. Zhao, F. Scholer, and Y. Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008.*, volume 4956 of *Lecture Notes in Computer Science*, pages 52–64. Springer, 2008.

[29] Y. Zhou and W. B. Croft. Ranking robustness: a novel framework to predict query performance. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 567–574, New York, NY, USA, 2006. ACM.