Exploring Reductions for Long Web Queries

Report by : Razvan Belet, Andreas Sander

1. Introduction

Analysis of major search engines query logs has been showing that long queries represent an increasingly higher fraction of the queries submitted to a web search engine. For instance, queries of length five words or larger have increased at a year rate of 10%. This increasing trend is also determined by the question answering field which is centered on long queries.

Broadly, there are 2 main types of query reduction techniques: query reweighting and query reduction. Both of them 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.

In query re-weighting, each term of the query is given a certain weight which represents its “importance” in the query. More important terms are given larger weights. The way these weights are obtained is a research topic by itself and it is beyond the purpose of this report. 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.

In query reduction, terms or combination of terms are completely dropped from the original query and the resulting shorter versions replace the original query (in case they perform better). Given this, query reduction requires analyzing a potentially exponential number of reduced versions of the original query which implies the usage of exponential complexity algorithms. On the other hand, a very big advantage of this technique is that it can be seamlessly integrated with existing search engines, e.g. as a loosely-coupled layer on top of existing search engine implementations. At the same time query reduction relies heavily on query quality prediction measures like Clarity. Broadly, there are 2 main categories of query quality prediction techniques depending on the time when the prediction is done: pre-retrieval (prediction is done before executing the query) and post-retrieval (prediction is done after executing the query). As opposed to pre-retrieval techniques which naturally rely only on query dependent features, the post-retrieval techniques combine query-dependent features with

retrieved document features in order to predict the quality of a query. The number and the exact features to be used depend on the concrete implementation and they can be: BM25 scores, PageRank scores, click-troughs and boolean features such as whether the query contains stop words, URLs, location words. The post-retrieval approaches rely on language models obtained from the retrieved results and consequently they are expensive to compute. Besides this, it has been shown that the existing query quality prediction techniques perform well on TREC data but not on web data (the main reason behind this being the lack of homogeneity in web data as opposed to TREC data).

2. Query Reduction

In this section we describe the main contributions of the paper with the focus being on the proposed approach to web query reduction.

The main contributions of the paper consist in:

- Proposing a query reduction method applicable to web queries (existing reduction techniques are performing well on TREC data one characteristic of whom is the homogeneity. On the other hand web data is not homogenous which makes the existing techniques ineffective).

- Evaluating the proposed method on $\mathcal{P}^Q$ a large set of web queries extracted from the log a major search engine.

The problem of query reduction can be described as follows:
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 $Q = \{q_1, \ldots, q_n\}$, we use 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 follows:

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

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:

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

The paper introduces a method for computing $\widehat{T_f}(P)$. The prediction is done using a previously trained regressor as described below.
The approach can be summarized as follows: It is divided in 2 main phases. In the first phase (training phase) a regressor(h*) is trained using a consistent set of training queries({Q_i}) represented as sets of terms ($Q_i = \{t_{i,1}, t_{i,2}, \dots, t_{i,n}\}$). In the second phase (prediction phase) the performance of a new incoming query is predicted as follows:

1. Create the set of reduced queries including the original query : $P^Q = \{Q, ReducedQuery_1, \dots, ReducedQuery_k\}$
2. Predict the performance of each member of P^Q using regressor h*
3. Choose the query with the best predicted performance and return its results

The workflow described above is straightforward. However, there are 2 aspects which differentiate the method from the rest. The first aspect worth noticing is that the method is not exponentially complex because it does not take into consideration all the possible reduced versions of the original query as most of the similar approaches do. Instead, it only takes into consideration the reduced versions which are obtained by dropping one term. Intuitively, this strategy turns the method into an approximated one (according to the experiments the method delivers very good results, so the approximation is a reasonable one). One of the reasons for which this approximation works is that the pruned candidates (i.e. the queries obtained by dropping more terms) are not strong candidates anyway so pruning them does not lead to significant loss of precision.
A second aspect which needs to be mentioned is that the approach proposes 3 types of regressors or, as the authors call them, 3 different learning formulations of the original problem formulation. All of the 3 learning formulations described below transform the problem of query reduction into a query performance prediction problem:

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.:

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

2. *Difference Prediction*: In this 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. We learn a least-squared errors regressor h* 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:

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

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 queries. The pairwise preferences induce a partial ordering and the query at the top of the ordering is selected.

We notice that computing the regressors is the core task here. The paper proposes using Random Forests in order to compute the *Independent*, *Difference* regressors and RankSVM to compute the *Ranking* regressor. The features used to build the Random Forests are divided in 2 categories: Query features which refer only to characteristics of the query (lexical features flagging the presence of URL, stop words, or location words as well as query length) and Query-document features (BM25 scores, PageRank scores, LR scores, click-through counts).

3. Experimental Results

All 3 regressors are trained using a number of around 12,000 queries extracted from the logs of a major search engine. The evaluation data consists in 6400 long queries frequency-weighted sampled from the Web search engine. For each long query all the reduced queries (by dropping 1 term) are obtained. Each query is executed and afterwards human annotators are asked to judge the relevance of all queries (reduced & original) with respect to the original query.

For each formulation, the paper reports results for 2 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 depend on the sign of the difference between their predicted NDCG@5.

Query Replacement

Table 1 shows the performance of the different problem formulations. The paper compares the 3 formulations using 2 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.

	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%)

Table 1

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. 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.

Results Interleaving

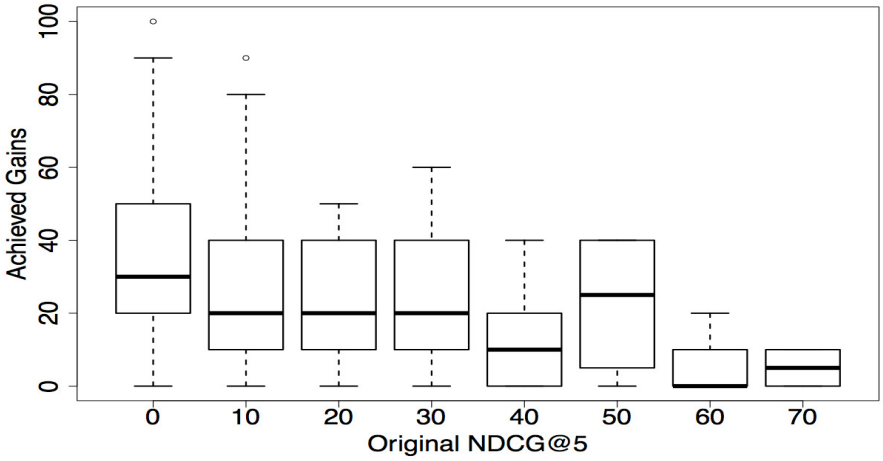
Table 2 shows the gains achieved by the interleaving results. Difference achieves the best overall gains, whereas Independent achieves the best subset gains. Difference and Ranking both have a positive impact on a large number of queries, 31% and 25% respectively, whereas Independent provides positive gains for only 4%. 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.

	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%)

Table 2

4. Strengths, Weaknesses and Extensions

One of the main strenghts of the paper is that the proposed approach is a query reduction method which works well for web queries. As previously mentioned there are many existing query reduction methods but none of them is effective in the context of web queries mainly because they rely on the homogeneity of the data collection to be queried.
A second siginificant strength of the approach presented in the paper is that it can be seamleassly integrated into an existing search engine and extend its functionality with the query reduction feature. The way the method works, allows one to integrate the query reduction functionality as a layer on top of an existing search engine.
Yet another strength of the query reduction method presented in the paper is the linear complexity. As opposed to many other similar techniques, this method does not take into consideration all the possible reduced versions of an original long query (which would lead to exponential complexity) but only those obtained by dropping 1 term and consequently the complexity of the method is a linear one.
Last but not least, the method provides significant query quality improvements for originally bad performing queries (see figure below). In other words, the method improves where there is room for improvement. Moreover with the results interleaving method (where the results from of the original and reduced query are interleaved) the risk of a worse result quality, due to an erroneous choice of a query, is minimized.



As already mentioned the approach also has some weaknesses. The described approach evaluates for every web query whether there is a better performing reduced query version of it. This can be really burdensome for a search engine.
Therefore the approach must be modified to increase the efficiency-performance ratio. On the figure above we can see that the NDCG@5 gain is the significantly higher for low-performance original queries. Hence, increasing the efficieny-performance ratio means evaluating the performance of the reduced web queries only for original queries which perform worse in NDCG@5. In order to do that, one possibility could be to choose a threshold for the predicted performance of the original query. For example the evaluation of reduced queries would only start if the predicted performance of the original query is below a threshold of 25 points on NDCG@5. The negative side-effect of this approach is that the performance of the original query still needs to be predicted for every issued query. Another approach could be to take a closer look to the training data, compute several statistical measures (mean, standard deviation) according to the length(number of words) of the queries and take a decision for a length threshold. With this approach the performance prediction of the original query and a possible evaluation of the reduced queries would only start if the original query exceeds a specific length.

5. References

[1] Niranjan Balasubramanian, Giridhar Kumaran and Vitor R. Carvalho. Exploring Reductions for Long Web Queries.

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

[3] G. Kumaran and V. Carvalho. Reducing long queries using query quality predictors.

[4] C. Hauff, V. Murdock, and R. Baeza-Yates. Improved query difficulty prediction for the web.

[5] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors

[6] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log