A Query Substitution-Search Result Refinement Approach for Long Query Web Searches

Yan Chen and Yan-Qing Zhang
Department of Computer Science
Georgia State University
Atlanta, U.S.A.
ychen44@student.gsu.edu, yzhang@cs.gsu.edu

Abstract—Long queries are widely used in current Web applications, such as literature searches, news searches, etc. However, since long queries are frequently expressed as natural language texts but not keywords, the current keywords-based search engines, like GOOGLE, perform worse with long queries than with short ones. This paper proposes a query substitution and search result refinement approach for long query Web searches. First, we retrieved several short queries related to a long query from the users’ query history. Then, we constructed the short query clusters and selected the most representative queries to substitute the original long query. However, since searching relevant short queries may ignore contexts and terms in the original long query and thus obtain diverse results and neighboring information, we compared the contexts from search results with the contexts from original long query and filtered non-relevant results. The experiments show that our approach achieves high precision for long query Web searches.

Keywords-long queries; web searches; short queries

I. INTRODUCTION

Long queries, as an important query type among the searches on the Web, play a crucial role in many information search applications, such as literature searches, news searches, or searches in environments where queries are expressed as natural language texts but not keywords. The purpose of long query searches is to identify the most relevant existing information, like articles, news or Web pages, to the contents of long queries. Although long queries are more convenient to express complex and specific information needs than the keyword-based queries, current commercial search engines, in general, perform worse with long queries than with short ones. The major reason is that most search engines respond to a user’s query by using the Bag-of-Words model [12], which assumes queries and Web documents are composed of individual words neither related nor ordered. However, for long natural language queries, tokens are ordered words and phrases with underlying semantic relationships. Thus, a lack of sufficient natural language parsing causes search engines not to understand semantic queries [7]. Also, since traditional search engines treat all terms from the users’ inputs equally, they lose focus on the key concepts that have the most impact on the retrieval results [13]. Moreover, noise or redundant terms will further degrade the effectiveness of long query Web searches if searching all query terms equally [6]. Thus, users frequently have to convert their long natural language queries into a few adequate terms for retrieving precise search results, which increases users’ burdens.

Query reduction is a technique that eliminates noisy and redundant terms, and uses underlying retrieving models to extract key concepts from long queries. Formulating a shorter query from the original long query has been proven to lead to significant improvements in Web search applications [6]. Also, searching key concepts from long queries causes adequate search results since using long queries has a higher chance of retrieving fewer search results. However, identifying key concepts in long queries is still a challenge task because of its heavy reliance on corpus statistics. Moreover, since query reduction techniques remove context from the original query, it leads to a loss of specificity of the original query and may seek more diverse information than the original query.

Query segmentation is a technique that segments queries into concepts, and thus search engines retrieve Web documents based on the concepts but not tokens [10]. Jones et al. proposed a mutual information based approach to determine segmentation breaks between pairs of tokens [14]. Tan and Peng [1] proposed an unsupervised machine learning approach to discover queries’ underlying concepts based on a generative language model, and used expectation-maximization (EM) algorithm to estimate the model’s parameters. Also, they incorporated Wikipedia to enhance the unsupervised machine learning. Since their approach identifies key concepts of queries, it greatly improves the retrieval performance for long queries. However, as discussed in the previous paragraphs, because query segmentation method treats all query concepts equally and loses focus on the key concepts, it degrades the effectiveness of long query Web searches.

This paper presents a Query Substitution-Search Result Refinement approach to improve the performance and effectiveness of long query Web searches. For a long query, we do not have to search the documents that contain exact texts or sentences as the long query has, but retrieve the documents containing the topics that the long query includes. Based on this observation, we proposed the query-
substitution strategy that replaces the current long natural language query by past relevant short keywords-based queries. Since current search engines generally perform worse with long queries than with short keywords-based queries, searching relevant short queries may improve the retrieval performance and provide approximate search results as searching the original long query. Also, the past short queries may provide us with what specific aspects interest users, and we may discover the users’ interesting topics in a given long query. Moreover, since the short queries are created by users, they should contain the significant and meaningful terms for describing the topics. In contrast with our approach, the query reduction approach may create non-meaningful queries. Although the query substitution strategy may improve the effectiveness of long query Web searches, we may still obtain diverse results and neighboring information since searching relevant short queries may ignore contexts and terms in the original long query. Thus, we proposed the search result refinement strategy that filters non-relevant results by evaluating the similarities of contexts from results and contexts from the original long query.

The major contributions of this paper are:
2. For the long queries containing several topics, our clustering-based short queries substitution significantly improves the performance of Web searches.

This article is organized as follows: Section 2 presents a brief overview of the related work. Our proposed method for query substitution is described in Section 3. The methods for search result refinement are described in the Section 4. Data collection and experiments are presented in Section 5. Finally, we address the conclusions and future works in Section 6.

II. RELATED WORKS

Jones et al proposed a query substitution approach that generates a modified query based on past similar queries or phrases, and replace the original query by the modified query [14]. They built a machine learning-based model to select the appropriate query candidates based on their features relating to the original query and human judgments. Bonchi et al proposed a query decomposition approach that generates queries whose union of search results roughly matches the results of the original query [5]. They first instantiated the query decomposition problem to the set cover problem, and then applied agglomerative clustering algorithm and dynamic programming to solve the problem. Since the above two approaches replace one query by relevant queries, their strategies are pretty similar to ours. However, their approaches just deal with keywords-based short query substitution but not long natural language query substitution. Also, replacing one query by relevant queries may lose contexts in the original query, so their approaches may obtain more non-relevant results than our approach.

Since query reduction is also an effective approach for long query Web searches, we introduce the following well-known query reduction methods. Kumaran and Allan [6] have proposed a query iterative reduction technique that uses a small subset of concepts or entities from the original query to substitute the original query and let users determine the most appropriate combinations of term subsets. Their method obtains significant performance improvement on TREC queries. However, this technique is difficult to realize in practice due to the exponential number of options that need to be analyzed. Also, this method makes users’ burdens heavier. Later on, Kumaran and Allan [7] proposed a selective interactive query reduction approach that determines appropriate query reductions based on users’ implicit feedback. Their approach reduces the users’ time and effort for interacting with the query reductions. Shapiro and Taka [11] proposed a method that extracts intersecting terms from the long queries as sub-queries and merges the results from searching those sub-queries. Zukerman, Raskutti and Wen [9] used decision graphs to identify the noisy attributes of query terms like syntactic, paraphrase-based and frequency-based attributes from TREC queries, and remove terms with these attributes in long queries.

III. SUBSTITUTING A LONG QUERY BY RELEVANT SHORT QUERIES

In this section, we introduce our approaches for selecting short queries related to a long query and building a new query based on these relevant short queries.

A. Constructing a Small Working Set of Short Queries for a Given Long Query

The clickthrough data sets that contain users’ queries and corresponding clicked URLs are used as data source for collecting relevant short queries. Since users’ clickthrough data sets commonly contain a huge number of queries, we need to construct an initial query set IQ including a small number of short queries that are potentially related to a given long query. In order to evaluate the similarity between queries, we applied the frequency of terms occurring in both queries to measure their relations. Since long queries are natural language texts and may contain multiple sentences, we applied the sentence-query similarity to calculate the relevance score of a long query and a short query [4].

\[
R(LQ, SQ) = \sum_{i=1}^{n} \left( \frac{|S_i \cap SQ|}{|SQ|} \right) \tag{1}
\]

For the above formula, \(R(LQ, SQ)\) represents the relevance score of the long query \(LQ\) and the short query \(SQ\); \(S_i\) denotes the \(i^{th}\) sentence in the long query \(LQ\); \(|S_i \cap SQ|\)
denotes the count of common terms occurring in \(S_i\) and \(SQ\); \(|SQ|\) represents the number of terms in \(SQ\).

Based on the relevance scores of the long query and short queries, we rank the short queries and put top \(m\) short queries into the initial query set \(IQ = \{q_1, q_2, \ldots, q_m\}\). Since some short queries may contain key concepts of the long query but others may just contain noise information, we have to determine which short queries are significant for the long query in the set \(IQ\).

### B. Collecting Highly Related Short Queries from the Working Set for the Long Query

For a short query, if a user clicks a corresponding result page, the sentences around the query terms in the clicked result page should contain users’ interesting information. Based on this observation, we may propose the following two assumptions.

**Assumption 1.** The short query represents the key contents of the sentences around the short query terms in the clicked result page.

**Assumption 2.** A short query and the long query are related if the context of the sentences around the short query terms in the clicked result page and the context of the long query are similar.

Based on those assumptions, we propose the following methods to calculate the contexts relations. First, we extract the noun phrases from the set of sentences \(ST = \{s_1, s_2, \ldots, s_t\}\) around the short query terms and the noun phrases from the long query. Then, the relations of the contexts of sentences in \(ST\) and the context of the long query will be determined by the number of common noun phrases they have.

Since the set of sentences \(ST\) and the long query are natural language texts, we need to determine the noun phrases boundaries like “(New York) (Art Center)” in the phrase “New York Art Center”; identify name entities like the names of persons, organizations, locations, etc.; and determine whether two entities refer to each other like “John Kennedy” and “the president Kennedy.” We applied a machine learning- based noun phrase identification module [2] to recognize fine-grained name entities in the long query and the sentences in \(ST\). Also, we adopt a corpus-based, machine learning approach [15] for noun phrase coreference resolution, which resolve a certain type of noun phrase (e.g., pronouns) and also general noun phrases. Once co-referred entities are resolved, for a given referred entity \(en\), we replace all other referring entities to \(en\) in the long query and the sentences in \(ST\).

Once noun phrases boundaries are identified and co-referred entities are resolved, we extract noun phrases from the long query and the sentences in \(ST\) as their features. Then, we propose the following equation to calculate the context relations of the long query \(LQ\) and the set of sentences \(ST\) around the short query \(SQ\).

\[
R(LQF, STF) = \frac{2 \times |LQF \cap STF|}{|LQF| + |STF|}
\]  

(2)

In the formula (2), \(LQF\) denotes the features from the long query; \(STF\) denotes the features from the sentences in \(ST\); \(|LQF \cap STF|\) denotes the count of common features occurring in \(LQF\) and \(STF\).

Based on the relations of the long query \(LQ\) and the set of sentences \(ST\), we rank all the short queries in the query set \(IQ\) obtained in section 3.1; select top \(n\) short queries as the most relevant queries to the original long query; and put top \(n\) short queries into a relevant query set \(RQ = \{q_1, q_2, \ldots, q_n\}\).

Since a long query may contain several topics, its relevant short queries may correspond to different topics of the long query. For example, for a query “Tropical Storm Fay, stalled near Cape Canaveral, Florida, soaked portions of east-central Florida late Wednesday. Florida Gov. Charlie Crist has asked President Bush to declare an emergency in the state to free up federal funding”, some relevant short queries may refer to the topic of “Fay stalled Florida” but others may refer to the topic of “governor declare an emergency.” Thus, we may construct clusters of queries in \(RQ\) and select the most representative queries from clusters as the substitutions of the original long query \(LQ\).

### C. Building the Long Query’s Substitution

In section 3.2, we build a relevant query set \(RQ\) for the long query \(LQ\). Since a short query in \(RQ\) may be included by several sentences in \(ST\) and each sentence may contain several common features in \(|LQF \cap STF|\), we may construct a bipartite graph that connects the short queries in \(RQ\) and the features in \(LQF\).

Fig. 1 shows a short queries-long query features bipartite graph, where squares represent features and rounds represent queries. Then, we propose the following bipartite agglomerative clustering algorithm [3] to construct short query clusters based on the long query’s features they connect.

The threshold value \(\delta\) in the following algorithm one is a control factor for determining the sizes of clusters. A small value of \(\delta\) indicates that the queries in clusters have high relations but a big number of clusters may be created; a big value of \(\delta\) indicates inverse results. In our experiments, we pre-set a reasonable value of \(\delta\) for obtaining optimal results.
Algorithm 1 – Constructing query clusters

Input: A short queries-long query features bipartite graph \( G \)
Output: A clustered short queries-long query features bipartite graph \( G_c \)

1: Calculate the number of common features between all possible pairs of queries.
2: Merge two queries \( A \) and \( B \) that connect the biggest number of common features.
3: Calculate the number of common queries between all possible pairs of features.
4: Merge pairs of features connected by the biggest number of common queries.
5: Repeat the step 1 through 4 until the number of common features of any pairs of queries is smaller than a threshold value \( \delta \), or all queries are merged into one query.

Fig. 2 shows a process of constructing clusters of short queries based on the algorithm 1. In the Fig. 2 (a), since \( q_1 \) and \( q_2 \) are connected by the biggest number of common features \( f_2 \) and \( f_3 \), we merge \( q_1 \) and \( q_2 \). Then, we connect the new created query \( \{q_1, q_2\} \) with the features connected by \( q_1 \) or \( q_2 \) as shown in the Fig.2 (b). Since features \( f_4 \) and \( f_5 \) are connected by the biggest number of common queries \( \{q_1, q_2\} \) and \( q_3 \), we merge \( f_4 \) and \( f_5 \) and connect the new created feature \( \{f_4, f_5\} \) with the \( \{q_1, q_2\} \) and \( q_3 \). We iteratively merge queries and features, and finally obtain two query sets \( \{q_1, q_2, q_3\} \) and \( q_4 \).

After clusters of relevant short queries are created, we applied the TF-IDF weight [16] to select the most representative query from each cluster and put them together as a new query. Assuming that we have several clusters of queries \( C = \{c_1, c_2, \ldots, c_m\} \) and a query \( q \) from the cluster \( c_j \), we propose the following equation to calculate the query \( q \)'s score.

\[
S(q) = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{\sum_{t \in c_j} \text{num}_j(t)}{\sum_{t \in c_j} \text{num}_j(t)} \times \log \frac{|C|}{|\{c : t_i \in c\}|} \right)
\]  

(3)

In the equation (4), \( n \) denotes the number of terms that \( q \) contains; \( \text{num}_j(t_j) \) denotes the number of occurrence of term \( t_j \) in the cluster \( c_j \); \( \sum_{t \in c_j} \text{num}_j(t) \) the sum of number of occurrence of all terms in \( c_j \); \( |C| \) the number of clusters; and \( |\{c : t_i \in c\}| \) the number of clusters containing term \( t_i \).

For each cluster, we select the query that has highest score. Then, the selected queries from clusters are combined together as the substitution of the original long query. The new created query will be submitted to commercial search engines for retrieving relevant documents.

IV. WEB PAGES RESULTS REFINEMENT

Although the new created query contains the important features or topics of the original long query, we may still obtain non-relevant results since the new query may ignore some contexts of the original one. Thus, we proposed the following approach that filters non-relevant results by measuring the similarities of search results and the original long query.

First, in the search results, we identify the sentences containing the terms of the new created query. Second, we applied the machine learning-based noun phrase identification module used in section 3.2 to recognize entities or noun phrases in these sentences. Third, we proposed the equation (4) to calculate the relevance scores of search results and the original query. Then, the search results will be ranked based on the relevance scores and only top results will be returned to users.

\[
S(p_i) = \sum_{j=1}^{n} \left( \frac{|f_i \cap F(t_j)|}{|F(t_j)|} \right)^{1/k}
\]

(4)

In the equation (4), \( p_i \) denotes the \( i^{th} \) search result; \( f_i \) denotes the features extracted from \( p_i \); \( t_j \) denotes the \( j^{th} \) topic; \( F(t_j) \) denotes the features related to topic \( t_j \) in the original long query; and constant \( k \) is a control factor to determine the impact of relatedness of features \( f_i \) and \( F(t_j) \). As \( k \) increases, the \( p_i \) may obtain high score if several common features exist between \( f_i \) and \( F(t_j) \).
V. Experiments

The experiments are presented in this section. First, we constructed our data sets based on the TREC Robust 2004 data and three month query history of a commercial search engine. Then, we compared our approach’s performance with other two well-known approaches.

A. Data Collection

We constructed the long query test sets based on the TREC Robust 2004 test file, which contain 250 titles, descriptions and narrative sections. Since the descriptions are natural language texts and express the information needs, the descriptions of TREC Robust 2004 are considered as the long queries in our experiments. Since the titles express the same information needs as the descriptions and they commonly contain a few keywords, the titles will be considered as short queries in our test sets. In our experiments, we take titles as baseline for comparing the performance with our approach. The narrative sections are the lengthy part and express important attributes of search results. Thus, we may consider the narrative sections as measures to evaluate whether retrieved documents meet the information needs. Since the descriptions of TREC Robust 2004 always contain one topic, we select 100 descriptions of TREC Robust 2004 as the single-topic long queries. Also, we manually create 100 multi-topics long queries by combining the descriptions with related topics from the narratives sections of TREC Robust 2004. We construct two sets of long query testing data. The first test set contains all single-topic long queries. The second test set contains all single-topic and multi-topics long queries. However, we divide the queries in second test set to ten groups. The first group contained 20 single-topic queries and 0 multi-topics queries. From the second group, each following group contained two more multi-topics queries and two less single-topic queries than the previous group. Therefore, the tenth group contained twenty multi-topics queries and no single-topic queries.

We constructed our short query data sets based on the log data of a commercial search engine. The log data consist of more than 20M Web queries from 650k users over three months, from March 1, 2006 to May 31, 2006. The number of clicked URLs for the 20M Web queries is 19,442,629.

For the query collection, first, we filtered the queries by only keeping the queries that were written in English and only contained alphabet characters and spaces. Second, we only preserved the queries that contain less than three terms as short queries. Third, we preserved the queries that resulted in at least one click per session. Since it was impossible to ask the original users to evaluate the search results’ quality, we assumed all the clicked Web pages containing the information that the users needed. Thus, based on the Assumption 1 in section 3.2, the queries should represent the key contents of the sentences around the query terms in the clicked result pages. Then, for our long query test data, we applied equation (1) and (2) to retrieve no more than twenty potential relevant short queries from our short query sets as our working sets.

B. Experiment Design

Parameters Setting. Since our initial experiments showed that a large set of short query candidates may contain noise information irrelevant to the original long query, we selected no more than five relevant short queries for a long query.

Since our test sets contain one hundred single-topic and one hundred multi-topics long queries, and the average number of topics of long query test set is 1.7, we selected a reasonable value of parameter δ for constructing clusters of relevant queries in the algorithm one. Based on our initial experiments results, the value range of δ should be between 1 and 3 so that we may create approximate number of clusters as the number of topics in the test sets.

In order to enlarge the impact of relatedness of features extracted from result pages and features from the original long query, we set parameter k=2 in the equation (4).

Comparative Approaches. We used the titles of TREC Robust as baselines to compare the performance with our approach. Also, we compared our approach’s performance with the following two well known methods.

Concept Weighting. Bendersky and Croft [13] proposed a machine learning-based concepts weighting approach to identify key concepts in verbose queries. They constructed vectors containing seven weighting features to represent concepts. Then, for the training set, they seek to learn a ranking function so that the rank of concept i is higher than concept j if the function value of concept i’s feature vector is larger than concept j. Once the learning process complete, they selected key concepts from verbose queries based on their ranks.

Query Reduction. Shapiro and Taksa [11] proposed an approach that contains following steps for query reduction: removing stop words from the original query and decomposing the original queries into several search terms; ranking those search terms based on the TF-IDF Weight; generating a fixed/variable number of sub-queries by randomly selecting terms from the search terms; submitting sub-queries as inputs to a search engine and merging the search results from those sub-queries into one rank list.

Evaluation Measures. Two measures, P@10 and MAP, are used for comparing our approach’s performance with others. P@10 is the precision of top 10 rank documents returned by a search engine. MAP [8] is the arithmetic mean of average precisions (AP) of a set of queries and AP is the sum of the precision at each relevant document in the search results divided by the total number of relevant documents in the collection.
Correct Results for Testing Data. Since we did not have a standard result set to evaluate the correctness of the search results of long query Web searches, we had to manually speculate the correctness of search results. In order to minimize the bias for the results speculation, we collected the search results from all approaches; constructed indexes for search results and put indexes in a separate file; put all results in a corpus and justified their correctness. Thus, before justifying search results, we did not know which approach creates them, so this approach minimizes justification bias.

C. Experiment Results

Our main hypothesis is that substituting a long query by relevant short queries and then filtering non-relevant search results is more beneficial than machine learning-based key concepts identification or query reduction. Also, compared to other approaches, our approach’s performance is less impacted by the number of multi-topics long queries. Thus, in the following experiments, we compare the performance of our approach with other approaches on the test set one and two separately.

Table 1 shows the comparison results of three approaches by P@10 and MAP on the test set one. Baseline and the query reduction approach perform worse than concepts weighting and our approach, and our approach performance is slightly better than the concept weighting. In the baseline approach, queries are comprised by a few keywords and thus do not contain enough context to describe the information needs, the search results may be diverse. The query reduction approach creates sub-queries by randomly selecting terms from the original query, so a lot of non-meaningful or noise queries may be created. Thus, searching those sub-queries in parallel and combining their results may obtain numerous non-relevant results and greatly hurt the searching precisions. Although the concept weighting approach achieve better performance than query reduction and baseline approaches, its performance is heavily relied on corpus statistics. Since our approach selects the most representative queries to substitute the original long query, and filter non-relevant results, our approach significantly improve the searching precisions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Set 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P@10</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.235</td>
</tr>
<tr>
<td>Query Reduction</td>
<td>0.218</td>
</tr>
<tr>
<td>Concepts Weighting</td>
<td>0.314</td>
</tr>
<tr>
<td>Query Substitution</td>
<td>0.334</td>
</tr>
</tbody>
</table>

Figure 3. Comparison of different methods by P@10 on Test Set 2

Figure 4. Comparison of different methods by MAP on Test Set 2

We still used P@10 and MAP as evaluation metrics to compare three methods’ performance on the test set two. Since the test set two contains ten groups of long queries and the latter group contains more multi-topics queries than the previous group, we showed all approaches’ performance on these ten groups of test data in the Fig. 3 and Fig. 4. With the number of multi-topics queries increases, the trends of performance of these approaches can be observed.

As shown in the Fig. 3 and Fig. 4, with the growth of the number of multi-topics queries, the performance of the query reduction and concepts weighting approaches drops fast while our approach’s performance are not heavily impacted. Among three approaches, the performance of query reduction drops fastest since it randomly select terms from the original query to create new sub-queries. With the number of topics increases in the query, more non-meaningful or noise sub-queries will be created, and thus more non-relevant results will be retrieved. Since the concepts weighting approach identifies key concepts in single-topic queries instead of multi-topics queries, its multi-topics long query performance is not as well as the single-topic long query performance. Since our approach constructs clusters of short queries based on the topics they refer, and selects the most representative query from each cluster, our approach is less impacted by the multi-topics queries compared to other methods.
VI. CONCLUSIONS AND FUTURE WORKS

As a widely used query type, the current search engines do not perform well with long queries since the current search engines use the Bag-of-Words model while ignoring the semantics underlying queries. In this paper, we proposed a query substitution-search result refinement approach that first uses relevant short queries to replace the long query and then filter the non-relevant queries based on the context matching. The experiments prove that our approach performs better than concepts weighting and query reduction approaches. Also, for the multi-topics queries, our approach is less impacted than the above two approaches.

In the future, we will construct a Meta search engine that accepts long queries; substitutes the long queries by relevant short queries and submits the new created query to commercial search engines. Then, for the initial results returned by those search engines, we will filter non-relevant results based on the contexts and features they include. Then, based on users’ selection, we may judge the real performance of our approach.

REFERENCES