Improved Query Suggestion by Query Search

Xiaomin Zhang[1], Sandra Zilles[2], and Robert C. Holte[3]

[1] Amazon.com, `xiaominz@amazon.com`
[2] University of Regina, `zilles@cs.uregina.ca`
[3] University of Alberta, `holte@cs.ualberta.ca`

Abstract. At the Web Intelligence conference in 2009, Jiang, Zilles, and Holte introduced a novel approach to query suggestion based on query search (QSQS), as well as a system-centered evaluation method. For each potentially relevant document, QSQS creates a complex query—called a lexical alias for the document—that ranks the document in its top 20. A technique called Query Search then builds query suggestions by simplifying the lexical alias.

The present paper improves the state of the art by proposing two new query suggestion systems, IQSQS and GQSQS. Both replace the generation of lexical aliases by a simpler and more effective term selection process. They differ in their control structure: IQSQS builds query suggestions separately for each potentially relevant document, GQSQS builds them for a set of documents at once.

Both our new systems substantially outperform QSQS in the measures introduced by Jiang *et al.* to evaluate QSQS; we achieve improvements of up to 30 percent in these measures for short user queries and up to 100 percent for long user queries. We show empirically that query expansion, which forces the user's query to be included in each suggested query, is significantly superior to allowing the system the freedom to include or exclude terms from the user's query at its discretion.

1 Introduction

It is well known that users of search engines such as Google are unlikely to view documents beyond the top 20 returned by a query [15, 6, 5]. A query is therefore only effective in satisfying a user's information needs if relevant documents are returned in its top 20. If a user's initial query is not effective, it is necessary to issue subsequent queries until an effective one is found. Query suggestion systems assist the user in this process by suggesting a small number of alternative queries that are likely to be effective.

In this paper we follow the approach to query suggestion pioneered by Jiang *et al.* [9, 7] and present two query suggestion systems, called IQSQS and GQSQS, that are substantially superior to Jiang *et al.*'s system according to their own evaluation measures (called MCC and MEC) and methodology. These systems are our paper's main contributions. An additional contribution is strong experimental evidence that query suggestion systems aiming to score well according to MCC and MEC should do query "expansion", *i.e.*, they should add terms to the user's query rather than creating queries that do not contain the user's query.

2 Query Suggestion by Query Search (QSQS)

Following Jiang *et al.*, we say that a query "covers" a document if the document is among the top 20 documents returned by the search engine when the query is issued. Jiang *et al.*'s approach to query suggestion is based on their observation [8] that the probability of a document being relevant to the user's initial query is inversely proportional to the rank of the document in the initial query's results list. A query suggestion that covers documents that the initial query ranks high (but not in its top 20, since the user has already seen and rejected those) is therefore more likely to be effective than a query suggestion that covers documents that the initial query ranks low. They called the documents returned in positions 21–120 by the initial query "reference documents" and designed their query suggestion system to find queries that cover these documents.

2.1 Objective and Evaluation

Jiang *et al.* [9] evaluated a set of query suggestions by the number of reference documents the set covers, either collectively ("MCC") or on average individually ("MEC").[4]

For example, if the set of query suggestions contains 10 queries that collectively cover a total of 75 reference documents, MCC for this set of suggestions would be 75. If each of the suggested queries, considered individually, covered 8 reference documents, MEC would be 8.0. In general, MCC can be no larger than 100, the number of reference documents. The maximum possible value for MEC is 20, the largest number of reference documents a single query can potentially cover. For a given MEC value, the maximum possible value for MCC when 10 queries are suggested is 10*MEC. This happens only if no reference document is covered by two different suggested queries. The difference between MCC and this maximum value is an indication of the overlap in the set of reference documents covered by different suggested queries. In the extreme case, when exactly the same set of reference documents is covered by each of the suggested queries, MCC will equal MEC.

MCC and MEC relate to the standard Recall measure in the following way. Recall is the ratio of the number of retrieved relevant documents over the number of all relevant documents. In our context, all and only the reference documents are considered relevant. Since their number is always 100, it can be ignored. Retrieving a document by a query in our context means covering it, *i.e.*, returning it among the top 20 results. Hence MCC measures the collective Recall of the set of query suggestions a system returns, whereas MEC measures the expected Recall of a single query suggestion, averaged over all query suggestions returned for one initial query.

In a similar way, it would be possible to compute Precision values for Jiang *et al.*'s system and our new systems. High Precision values are not what we strive for though, for the following reason. Achieving high Precision means that the query suggestions made by the system cover as few non-reference documents as possible. As opposed to that, one of our declared goals is to have a substantial number of non-reference documents among the results covered by a query suggestion, in order to allow for a more

[4] This is the simpler pair of measures Jiang *et al.* proposed. The other measures take into account the exact ranks of the reference documents covered. "MCC" is an abbreviation for "Measure of Cumulative Coverage", "MEC" means "Measure of Expected Coverage".

diverse set of documents to be displayed to the user and to take into account the fact that the assumption that *only* the reference documents are relevant is not realistic. We hence restrict the evaluation of our systems to the MCC and MEC measures.

The aim of the system developed by Jiang *et al.* [9], which we call Query Suggestion by Query Search (QSQS), is to find a set of 10 suggested queries that maximizes MCC and, as a secondary criterion, also maximizes MEC.

2.2 The QSQS System Architecture

The design of the QSQS system closely follows the design of Martin and Holte's system for finding content-based addresses for web documents [12] and makes heavy use of their "Query Search" method. Query Search is a generic method for finding a query that covers a document by forming queries from various subsets of a given set of search terms and testing if the document is covered by issuing those queries to the search engine and examining their top 20 results.

Pseudocode describing the key steps in the QSQS system is given in Algorithm 1. The first processing stage (lines 1 to 4), called "Lexical Alias Search", constructs, for each reference document, a query, called the document's lexical alias, that covers the document. The second processing stage (lines 5 to 10), called "Query Suggestion Candidate Search", uses the lexical aliases to construct a set of queries from which the final query suggestions will be drawn. The third processing stage (lines 11 to 16) uses a greedy method to select the final set of K queries to suggest to the user.

Algorithm 1 Query Suggestion by Query Search (QSQS)

```
Input: a set RefDocs of reference documents and a number $K$
Output: a set $QS$ containing $K$ query suggestions
 1: // Lexical Alias Search
 2: for all $d \in RefDocs$ do
 3:     use Query Search to find a lexical alias for $d$, $LA_d$, based on $d$'s title terms and most
        frequent terms
 4: end for
 5: // Query Suggestion Candidate Search
 6: initialize $QSC$, the set of query suggestion candidates, to be empty.
 7: for all $d \in RefDocs$ do
 8:     use Query Search to find the set, $QSC_d$, of minimal subqueries of $LA_d$ that cover $d$
 9:     $QSC = QSC \cup QSC_d$
10: end for
11: // Greedy Selection of final Query Suggestions
12: initialize $QS$ to be empty.
13: for $i = 1$ to $K$ do
14:     add to $QS$ the query $qs \in QSC$ that most increases MCC (break ties to maximally
        increase MEC)
15:     remove $qs$ from $QSC$
16: end for
17: return $QS$
```

3 Improvements to QSQS

In this section we describe a system, IQSQS, that follows the same general pattern of processing as QSQS. The key difference is that lexical aliases are not used in IQSQS. The primary role of lexical aliases in QSQS is to supply a sequence of search terms that will be combined in various ways to create candidate query suggestions. Instead of finding a lexical alias for each document d, ISQS constructs an ordered set of search terms drawn from document d. These are chosen and ordered based on their ability to cover any of the reference documents, not just the document from which they are drawn. By focusing, from the outset, on overall coverage rather than the coverage of a single document, it is hoped that the set of candidate queries constructed from these terms will be much better than the candidate queries constructed from the lexical aliases.

Algorithm 2 Improved Query Selection by Query Search (IQSQS)

```
Input: the user's query, $Q_0$, the set RefDocs of reference documents for $Q_0$, a number $N \leq 20$,
    and a number $K$
Output: a set $QS$ containing $K$ query suggestions
 1: // Term Selection
 2: for all $d \in$ RefDocs do
 3:     quickly find a set, $F_d$, of up to 20 terms in $d$ that are likely to be useful in constructing
        queries with high coverage
 4:     score each term in $F_d$ according to its coverage when combined with $Q_0$ to form a query
 5:     sort $F_d$ (highest scoring term first) and delete all but the first $N$ terms
 6: end for
 7: // Query Suggestion Candidate Generation
 8: initialize $QSC$, the set of query suggestion candidates, to be empty.
 9: for all $d \in$ RefDocs do
10:     generate a set, $QSC_d$, of queries built from terms in $F_d$
11:     $QSC = QSC \cup QSC_d$
12: end for
13: // Greedy Selection of final Query Suggestions
14: initialize $QS$ to be empty.
15: for $i = 1$ to $K$ do
16:     add to $QS$ the query $qs \in QSC$ that most increases MCC (break ties to maximally
        increase MEC)
17:     remove $qs$ from $QSC$
18: end for
19: return $QS$
```

Pseudocode for IQSQS is shown in Algorithm 2. The first processing stage (lines 1 to 6), “Term Selection”, replaces the “Lexical Alias Search” stage in QSQS. The second processing stage (lines 7 to 12) serves exactly the same purpose as the “Query Suggestion Candidate Search” and is similar in many of its details. The third processing stage (lines 13 to 18) is identical to QSQS’s. We will now describe the first two stages of IQSQS in detail.

3.1 Term Selection

For each reference document d, the Term Selection stage has two steps: pre-selection (line 3 in Algorithm 2) and final selection (lines 4 and 5).

The input to the pre-selection step is the entire set of terms in the reference document, which may number in the thousands. Pre-selection reduces this number to around 20, a manageable number for the somewhat expensive scoring function used in the final selection step. Most term selection methods can be applied here, we examined two. The "Frequency" method selects the 20 terms that occur most frequently in the document; the "Snippet" method uses the terms in the fragment of text extracted from the document that Google returns to indicate the connection between the document and the user's initial query. An experimental comparison (not reported here, see [22] for details) showed that IQSQS's MCC and MEC scores with either of these methods were virtually the same. In the remainder of our experiments we used snippets for term pre-selection.

The final selection of terms involves scoring each term t individually by appending it to the user's initial query Q_0 and issuing the resulting query, which we will refer to as "$Q_0 + t$", to the Google API so that its coverage can be assessed. Our coverage score takes into account two factors, OC (Overall Coverage score) and LA (Lexical Alias score). For term t in the set F_d of terms for document d, the OC score is the number of reference documents that $Q_0 + t$ covers, and the LA score is 1 if $Q_0 + t$ covers d and 0 otherwise. These two scores are multiplied by weights and summed up to get the term's final score. In our experiments the weight for LA was three times the weight for OC.

3.2 Query Suggestion Candidate Generation

For each reference document, a small set of terms have now been selected and ordered according to the OC and LA scores. The next processing stage creates queries from these terms that will be the candidates for suggesting to the user. A set of candidates is created for each reference document using the selected terms for that document.

Although there are only a small number of terms to consider at a time, the number of possible queries that can be created from even as small a number of terms as 10 is astronomical. Every different subset is a different query, as is every different ordering of the terms. As we will see below, repeating a term in a query changes the results returned in the top 20 and therefore provides yet another way of defining queries from terms. Terms could also be combined into phrases (a sequence of terms surrounded by double quotes), or adorned with special directives (such as "+"), and so on. It is certainly possible to generate queries using the full range of options available, but in IQSQS, like QSQS, we have taken a very simple approach, only generating queries by taking subsets of the terms that have been selected for each reference document. In addition, we severely restrict the size of these subsets. The order of the terms in a query is always the order in which they occur in F_d.

We considered two ways of generating candidate queries from a given set F_d of terms. The first, AC, sets N, the number of selected terms, to 10, and generates all subsets (with no repetitions) of sizes 1 to 3, thus generating 820 (10+10*9+10*9*8) query suggestion candidates per reference document. The second method, BS, uses beam search to enable N to be larger; all pre-selected terms are considered instead of just 10.

	MCC	MEC
AC	56.66	7.26
BS	53.90	6.69
Q_0AC	70.88	8.99
Q_0BS	70.82	9.15

Table 1. Comparison of Query Suggestion Candidate Generation methods on short user queries.

BS first ranks all the pre-selected terms by their OC scores, using each individually as a length 1 query. Then, all length 2 queries that can be created by expanding one of the B top-scoring length 1 queries are ranked with respect to their OC score. B is called the "beam width"; it was 15 in our experiments. Finally, BS generates all the length 3 queries that can be created by expanding one of the B top-scoring length 2 queries. All queries of lengths 1–3 thus generated are considered as query suggestion candidates.

We are interested in whether there is any benefit for query suggestions to include the user's initial query Q_0 as part of the query suggestion; we therefore considered variations of AC and BS called Q_0AC and Q_0BS. The query suggestion candidates for Q_0AC are computed by taking each query suggestion candidate created by AC and appending it to Q_0. For Q_0BS the beam search generates potential query suggestions exactly as described above but it evaluates a query q via the OC score of $Q_0 + q$.

3.3 Experiment Comparing AC, BS, Q_0AC and Q_0BS

We compare the MCC and MEC scores of the query suggestions produced by AC, BS, Q_0AC and Q_0BS on 50 short user queries (length 2 or less) drawn at random from the 250 short queries used in Jiang *et al.*'s experiments [9, 7]. The average MCC and MEC scores over the 50 queries are shown in Table 1.

The most obvious conclusion from Table 1 is that including Q_0 in a query suggestion is of enormous benefit, increasing both MCC and MEC by approximately 25% regardless of whether AC or BS is used to generate query suggestion candidates. In all our results, the statistical significance of the difference in the scores (MCC or MEC) of two systems was determined using a sign test. Each of the 50 queries used in an experiment was considered an independent Bernoulli trial, with the null hypothesis being that it was equally likely, in any given trial, for either system to outperform the other. A difference was considered significant if the p-value computed in this way was less than 0.01 (p = the probability of the observed difference occurring by chance). The scores (MCC or MEC) of the system (AC or BS) with Q_0 used in the query are significantly better than the scores of the same system without Q_0 used in the query.

The difference in scores (MCC or MEC) between the system using AC and the system using BS are fairly small but statistically significant; AC outperforms BS on both measures. The MCC and MEC differences between Q_0AC and Q_0BS are not statistically significant. As a final comparison we ran these two systems on 50 long user queries (length 3 or more) drawn at random from the 250 long queries used in Jiang *et al.*'s experiments [9, 7]; the average MCC and MEC scores are shown in Table 2. The MCC and MEC differences between the two methods are statistically significant ($p < 0.001$). We

	MCC	**MEC**
Q_0AC	73.83	9.70
Q_0BS	78.05	10.66

Table 2. Comparison of Q_0AC and Q_0BS on long queries.

conclude that Q_0BS is the best of the Query Suggestion Candidate Generation methods we explored and use it in subsequent experiments involving IQSQS.

4 Greedy Query Suggestion by Query Search (GQSQS)

QSQS and IQSQS process each reference document individually to accumulate a set of query suggestion candidates and, at the very end, select K of them as query suggestions. In a second variant on Jiang *et al.*'s system, we change the control structure. Instead of generating query suggestion candidates for each reference document separately, we generate one query suggestion at a time from terms extracted from all reference documents, each time aiming for the largest possible MCC increase. The pseudocode of this system, Greedy Query Suggestion by Query Search (GQSQS), is shown in Algorithm 3.

GQSQS first identifies a set F of promising search terms in the same way IQSQS generates terms (line 3). The system then proceeds in K rounds, where K is the number of query suggestions to be produced. In each round, one query suggestion is generated in the following way. The first processing stage (lines 8 to 11), "Term Selection", selects terms from F in a way that is similar to IQSQS, except for a change in the coverage score measure, which takes into account the whole set of remaining (not yet covered) reference documents. Query suggestion candidates are generated (line 13) on the selected terms as in IQSQS, with Q_0BS. In each round the query that most increases MCC is chosen (breaking ties by selecting a query that also most increases MEC) and added to the set of final query suggestions (line 14). At the end of each round, we update the set of not yet covered reference documents (line 16) and, for each search term, update the set of covered documents accordingly (line 18).

The modified coverage score of a term t, used by GQSQS in the term selection stage, results from adding together the two following scores:

- the OC (Overall Coverage) score of t, as used in the term selection stage by IQSQS,
- the EOC (Extra Overall Coverage) score, for the current round index i, which equals the number of reference documents covered by $Q_0 + t$ but not yet covered by the query suggestions added to the set QS in rounds prior to round i.

Terms achieving the highest modified coverage score are ranked highest.

5 Comparison of QSQS, IQSQS, and GQSQS

To compare QSQS to our two new systems, we ran experiments using the same sets of 50 short queries and 50 long queries as used for the experiments reported in Section 3.

Algorithm 3 Greedy Query Suggestion by Query Search (GQSQS)

```
Input: the user's query, $Q_0$, the set *RefDocs* of reference documents for $Q_0$, a number $N \leq 20$,
    and a number $K$
Output: a set $QS$ containing $K$ query suggestions
 1: initialize $QS$, the set of query suggestions, to be empty
 2: initialize *DocsToCover*, the set of not yet covered reference documents, to equal *RefDocs*
 3: quickly find a set, $F$, of terms occurring in documents in *RefDocs* that are likely to be useful
    in constructing queries with high coverage score wrt *RefDocs*
 4: for each $t \in F$ do
 5:     *Covered*$(t)$ = set of documents in *RefDocs* that are covered by the query $Q_0 + t$
 6: end for
 7: for $i = 1$ to $K$ do
 8:     // Term Selection
 9:     using the size of *Covered*$(t)$, score each term in $F$ according to its modified coverage
        score wrt *DocsToCover* when combined with $Q_0$ to form a query
10:     sort $F$ (highest scoring term first)
11:     $F_i$ = the set of the first $N$ terms in $F$
12:     // Query Suggestion Generation
13:     generate a set, $QSC_i$, of queries built from terms in $F_i$
14:     add to $QS$ the query $qs_i \in QSC_i$ that most increases MCC (break ties to maximally
        increase MEC)
15:     // Update Set of Documents to be Covered
16:     remove the reference documents covered by $qs_i$ from *DocsToCover*
17:     for each $t \in F$ do do
18:         *Covered*$(t)$ = *Covered*$(t) \cap$ *DocsToCover*
19:     end for
20: end for
21: return $QS$
```

In these experiments, we used the Q_0BS method for generating query suggestion candidates in IQSQS and GQSQS.

The resulting average MCC and MEC scores for QSQS, IQSQS, and GQSQS are reported in Table 3. Our results show IQSQS and GQSQS superior to QSQS on both short queries and long queries. Sign tests (see Section 3) show the performance differences between IQSQS and QSQS (and between GQSQS and QSQS) to be highly significant ($p < 10^{-5}$ in every case). Hence we consider our new systems a substantial improvement over the state of the art. The MCC and MEC values achieved are noteworthy in their own right, not just in comparison with QSQS's. An MCC value over 67 means that over two-thirds of the reference documents are covered by one or more of the queries our systems suggest. An MEC value over 9 means that, on average, more than 9 of the top 20 documents retrieved by the each of the queries our systems suggest are reference documents, *i.e.*, highly likely to be relevant to the user's needs.

Most of the documents that our suggested queries retrieve that are not reference documents are novel documents, *i.e.*, not documents covered by the user's initial query. For IQSQS on the short queries, for example, of the 20 documents covered by one of our query suggestion, approximately 9 are reference documents (MEC=9.15, see Table 3),

	Short Query		**Long Query**	
System	MCC	MEC	MCC	MEC
QSQS	54.80	6.89	42.86	5.34
IQSQS	70.82	9.15	78.05	10.66
GQSQS	63.88	9.73	68.82	11.08

Table 3. Comparison of QSQA, IQSQS, and GQSQS, on short and long queries.

2 are documents covered by the user's original query, and the remaining 9 were ranked beyond position 120 by the initial query. Our suggested queries are therefore achieving a good balance between retrieving reference documents (very likely to be relevant given that the initial query's top 20 are not relevant), reminding the user of documents covered by the initial query, and injecting novelty into the set of results.

The observations that IQSQS outperforms GQSQS in terms of MCC, and that the opposite is true for MEC, are both highly significant statistically ($p < 0.005$ in both cases). In conclusion, both systems offer excellent performance in terms of both MCC and MEC. Applications that place greater emphasis on MCC should use IQSQS, and those that place greater emphasis on MEC should use GQSQS.

6 Query Suggestion Examples

Table 4 shows the queries suggested by Google, by QSQS, and by our methods IQSQS and GQSQS, for the queries "volcanos in italy", "herbs" and "ibm thinkpad 760c".

The most striking feature of Google's query suggestions are how "understandable" they are. It is very easy to imagine the subtopics they are intended to retrieve. However as the MCC and MEC scores show, these query suggestions are extremely poor at retrieving reference documents. Exactly the opposite is true of the queries suggested by our systems. They have relatively high MCC and MEC values, but in many cases it is not at all clear what subtopics they represent. We believe this is not a failing of our systems, or something that could be easily fixed by adding to our scoring criteria some measure of "understandability". We believe that the ranking functions used by Google, and undoubtedly other modern search engines too, have become sufficiently sophisticated and unintuitive that understandable query suggestions will often not be effective in satisfying a user's information needs.

A particular example of this phenomenon is the effect of repeating a term more than once; see Table 4. The effect on the documents returned in the top 20, like the effect of ordering the terms (which we observed in our work but did not systematically study), is substantial and largely unintuitive. Consider, for example, the query suggestion "herbs herbs" generated by IQSQS for the initial query "herbs". Here the reference document term "herbs" was appended to the initial query. IQSQS selects "herbs herbs" because this query covers more reference documents than other query suggestion candidates. In particular, the top 20 results for "herbs" are substantially different from those for "herbs herbs". Many documents covered by "herbs herbs" contain the term "herbs" twice in key positions such as the title. For instance, the titles of some top results for "herbs herbs" from Google (Nov. 24th, 2010) are "Herbs To Herbs", "Herbs Herbals herb and

Google	QSQS	IQSQS	GQSQS
major volcanoes in italy	italy volcanoes	volcanos in italy studies volcano volcanoes	volcanos in italy volcanos
famous volcanoes in italy	volcanos worldwide	volcanos in italy italy	volcanos in italy italy org
many volcanoes italy	volcano etna italy	volcanos in italy volcano erupted volcanoes	volcanos in italy pacific volcanos growing
three volcanoes italy	volcanoes italy active	volcanos in italy cams active east	volcanos in italy japan
	volcano diagram photo	volcanos in italy volcano feb 7	volcanos in italy lands
	italian volcanos	volcanos in italy tv etna	volcanos in italy 3350
	volcanoes forces nature mount	volcanos in italy moderate eruptions	volcanos in italy brief pacific world
	volcano information encyclopedia com	volcanos in italy explore eruption	volcanos in italy uploaded
	online volcano information volcanoes	volcanos in italy volcanos specifically	volcanos in italy deal
	amazon com volcano adventure guide	volcanos in italy pacific	volcanos in italy diagram
MCC=5 MEC=1.5	MCC=35 MEC=3.7	MCC=60 MEC=7.2	MCC=49 MEC=6.3
list of herbs	herbs herbal	herbs com herbs	herbs herbs learn herb
types of herbs	herbs website	herbs company herbs site	herbs herbs co
cooking herbs	herbs com	herbs herbal provides	herbs herbs com website
growing herbs	herbs herb gardens gardening	herbs herbs gardens	herbs information
culinary herbs	information herbs	herbs information	herbs medical herb site
pictures of herbs	herbs organic	herbs com vitamins	herbs herbs
medicinal herbs	herb store herbs herbal	herbs drying seeds method	herbs gardens
herbal medicine	medicinal herbs	herbs herbs chinese herbal	herbs herbs information database
	herbs home	herbs herbology 1	herbs remedies
	site herb growing herb herbal	herbs herbs education programs	herbs seeds
MCC=15 MEC=1.9	MCC=47 MEC=6.1	MCC=54 MEC=5.7	MCC=52 MEC=6.8
	thinkpad 760c replacement	ibm thinkpad 760c 755 760 ibm	ibm thinkpad 760c wholesale 760
	thinkpad 760c	ibm thinkpad 760c 365 760 ibm	ibm thinkpad 760c 560 365
	760c 9547	ibm thinkpad 760c 760 dont mailing	ibm thinkpad 760c car
	ibm centre thinkpad 755cv	ibm thinkpad 760c lcd 24	ibm thinkpad 760c 355
	memory ibm thinkpad 760c	ibm thinkpad 760c 9546 page laptop	ibm thinkpad 760c repair
	760c 760cd	ibm thinkpad 760c fix	ibm thinkpad 760c 29
	760c 9546 product	ibm thinkpad 760c 1995	ibm thinkpad 760c 370 shopping
	ibm thinkpad 760 reviews	ibm thinkpad 760c replacement 760ld	ibm thinkpad 760c shop
	thinkpad 760c win	ibm thinkpad 760c vista 760 ibm	ibm thinkpad 760c wholesale 755cd
	ibm 760c battery	ibm thinkpad 760c 760 image com	ibm thinkpad 760c 560e 755 380
MCC=0 MEC=0.0	MCC=60 MEC=8.2	MCC=80 MEC=11.0	MCC=63 MEC=11.4

Table 4. The query suggestions for the queries “volcanos in italy”, “herbs” and “ibm thinkpad 760c” by Google, QSQS, and our methods, as of Nov. 24, 2010.

herbal remedies – HerbsHerbals.com", "Herb's Herbs & Such", "Medicinal herbs – Affordable herbs", etc. These are not among the top results for the query "herbs".

If nowadays effective query suggestions necessarily border on being incomprehensible in terms of which subtopics they represent, research on query suggestion must pursue two goals. The first, represented by this paper, is to find ever better ways to create effective query suggestions without requiring that the queries be comprehensible. The second aim is to find comprehensible summaries of document sets, e.g., by cluster labelling methods [11, 16, 3, 2, 4, 17] or multi-document text summarization [10, 14].

7 Related Work

The approach to query suggestion introduced by Jiang *et al.* [9] of using Query Search to create query suggestions, is fundamentally different than other approaches because it evaluates the queries it creates by issuing them to the search engine and observing the documents returned in the top 20, rather than using a surrogate evaluation measure such as the similarity of the terms in the suggested query to those in the user's query [11, 1, 18, 19, 21]. Most similar to our approach is the *pseudo-relevance feedback* approach (also called blind-relevance feedback) [13, 20]. This assumes that the top T documents in the results of the user's query are relevant (including the top 20, unlike our approach) and extracts terms from these documents that best discriminate them from the documents not in the top T. These terms are used to construct query suggestions, but, unlike our approach, these suggestions are not evaluated by observing the results they return.

8 Conclusion

We proposed two new query suggestion systems using query search, based on Jiang *et al.*'s QSQS system [9]. The changes to QSQS consist mainly of replacing the construction of lexical aliases by a more elegant and more effective process of term selection. Query suggestion candidates are no longer generated by simplifying a complex query (a lexical alias) top-down, but by forming queries from promising search terms bottom-up. The two systems we present both use this method but vary in their control structure.

Both new systems substantially outperform QSQS in the measures that were proposed by Jiang *et al.* and that were explicitly used as objective functions in the design of QSQS. IQSQS improves QSQS by about 30% (both MCC and MEC) on short queries, and on long queries by about 80% (MCC) and 100% (MEC); GQSQS is even more effective in terms of MEC on long queries. Part of this improvement is due to forcing our systems to return queries that *expand* the initial query, as we verified empirically. This suggests that including the initial query in a query suggestion is generally advisable.

Acknowledgements. We thank Google's University Research Program for providing access to the Google API[5] and Shen Jiang for his help in early stages of this work. We gratefully acknowledge financial support by Google, the Alberta Innovates Centre for Machine Learning (AICML), and Canada's Natural Sciences and Engineering Research Council (NSERC).

[5] See http://research.google.com/university/search/docs.html for documentation.

References

1. C. Carpineto, R. Mori, G. Romano, and B. Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)*, 19:1–27, 2001.
2. J. Chen, O. R. Zaïane, and R. Goebel. An unsupervised approach to cluster web search results based on word sense communities. In *WI'08*, pages 725–729, 2008.
3. D. Cutting, D. Karger, J. Pederson, and J. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *ACM SIGIR 1992*, pages 318–329, 1992.
4. F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani. Cluster generation and labeling for web snippets: A fast, accurate hierarchical solution. *Internet Mathematics*, 3:413–443, 2006.
5. B. Jansen and A. Spink. How are we searching the world wide web?: a comparison of nine search engine transaction logs. *Inf. Process. Manage.*, 42(1):248–263, 2006.
6. B. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.*, 36(2):207–227, 2000.
7. S. Jiang. Searching for queries to improve document retrieval in web search. Master's thesis, University of Alberta, 2009.
8. S. Jiang, S. Zilles, and R. Holte. Empirical analysis of the rank distribution of relevant documents in web search. In *WI'08*, pages 208–213, 2008.
9. S. Jiang, S. Zilles, and R. Holte. Query suggestion by query search: a new approach to user support in web search. In *WI'09*, pages 679–684, 2009.
10. C.-Y. Lin and E. Hovy. From single to multi-document summarization. In *ACL*, pages 457–464, 2002.
11. C. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
12. J. Martin and R. Holte. Searching for content-based addresses on the world-wide web. In *Proceedings of the 3rd ACM Conference on Digital Libraries*, pages 299–300, 1998.
13. M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *ACM SIGIR 1998*, pages 206–214, 1998.
14. D. Radev, H. Jing, M. Sty, and D. Tam. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938, 2004.
15. C. Silverstein, M. Rauch Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.
16. B. Stein and S. M. Zu Eissen. Topic identification: framework and application. In *Proceedings of the International Conference on Knowledge Management*, pages 522–531, 2004.
17. P. Treeratpituk and J. Callan. Automatically labeling hierarchical clusters. In *Proceedings of the 2006 International Conference on Digital Government Research*, pages 167–176, 2006.
18. E. M. Voorhees. Query expansion using lexical-semantic relations. In *ACM SIGIR 1994*, pages 61–69, 1994.
19. X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 479–488, 2008.
20. R. White, C. Clarke, and S. Cucerzan. Comparing query logs and pseudo-relevance feedback for web-search query refinement. In *ACM SIGIR 2007*, pages 831–832, 2007.
21. J. Xu and W. Croft. Query expansion using local and global document analysis. In *ACM SIGIR 1996*, pages 4–11, 1996.
22. X. Zhang. Search term selection and document clustering for query suggestion. Master's thesis, University of Alberta, 2010.