# Query suggestion by query search: a new approach to user support in web search

Shen Jiang
*Department of Computing Science*
*University of Alberta*
*Edmonton, Alberta, Canada*
*sjiang1@cs.ualberta.ca*

Sandra Zilles
*Department of Computer Science*
*University of Regina*
*Regina, Saskatchewan, Canada*
*zilles@cs.uregina.ca*

Robert Holte
*Department of Computing Science*
*University of Alberta*
*Edmonton, Alberta, Canada*
*holte@cs.ualberta.ca*

***Abstract*—This paper introduces and analyzes a new approach to query suggestion. After the user issues a query $q_0$, for every document retrieved in a certain rank range $[\Theta_1, \Theta_2]$, a query search procedure constructs queries that rank the document high enough for the user to see it. From this set of queries the suggestions to be presented to the user are then selected so as to give the best possible access to the documents that were ranked in $[\Theta_1, \Theta_2]$ for the user's initial query $q_0$. This approach turns out to be successful under certain assumptions, which are discussed in this paper.**

## I. Introduction

Even information-literate users of web search engines often have difficulty formulating queries that allow them to easily find documents relevant to their information need. A user's query fails if all the relevant documents are so deep in its result list that the user will not reach them with the amount of scrolling that users are typically prepared to do. Often a query that fails can be turned into a successful query by simple modifications, such as adding terms, deleting terms, or replacing terms by synonyms or related terms.

Query suggestion is a technique that aims to improve the efficiency of a user's web search by suggesting alternatives to a user's initial query. This paper proposes and analyzes a new approach to query suggestion. The main idea is to construct a set of queries whose top-ranked documents are likely to be relevant to the user's initial query $q_0$ but are not top-ranked by $q_0$. Our approach is based on the assumption that some relevant documents occur reasonably high in the result list, just slightly too deep for the user to find them.

If we know which rank range the user usually views and in which rank range the relevant documents usually are, we can identify the documents that are likely to be relevant but not seen by the user. If we can find queries that would bring those documents into the rank range typically viewed by users, these queries can serve as query suggestions.

This idea motivates the following process. We employ a *query search* technique [9] to construct a search space consisting of potential query suggestion candidates and then search this space to identify "good" candidates (see Section III-A). After that, a query suggestion selection is performed to select from the candidates a certain number of query suggestions by maximizing an objective function.

This approach is only valid if we can verify that there actually are thresholds $\Theta_1$ and $\Theta_2$ such that most users are likely to view only the documents with a rank smaller than $\Theta_1$ in the result lists for queries they issue, and that a relevant document occurs in the top $\Theta_2$ documents of the result list with "sufficiently large" probability. A recent study based on transaction log analysis [5] actually gives evidence that $\Theta_1 = 21$ and $\Theta_2 = 120$ is a reasonable rank range for our approach. probability of about 65% that a relevant document is ranked in the top 120 results, with an increase of only about 2% when considering the top 300 results.

A system-centered evaluation approach is used to evaluate our system rather than a user study. Our evaluation method assumes that we know a likelihood function telling us at which ranks relevant documents are most likely to be found when a user issues a query. The function we use, from [5], may introduce some noise, but it can be applied without the expense of hand-labeled ground truth data.

The major contribution of this paper is our novel approach to query suggestion. This approach solves the query suggestion problem by suggesting queries that move up those documents that are likely to be relevant (ranked above $\Theta_2$) but likely to not be viewed by the users (ranked below $\Theta_1$). This makes our approach different from existing query suggestion approaches, *cf.* Section V. In our approach, we innovatively employ Martin and Holte's query search technique [9], which was invented for a different purpose, to generate query suggestion candidates. Our experiments verify that the candidates generated by query search are of good quality and therefore applicable in our approach and potentially other relevant applications.

Our second contribution is an evaluation method for query suggestion systems that is based on rank-relevance likelihood functions such as the ones in [5].

### A. Terminology

The following terminology is used throughout the paper, assuming a given web search engine (in our case Google).

We fix rank thresholds $\Theta_1$, $\Theta_2$ with $\Theta_1 < \Theta_2$. $\Theta_1$ is the expected rank below which users view documents in the result list returned by the search engine upon a query. $\Theta_2$ is a rank threshold up to which relevant documents occur in the result list with a "large enough" probability. We use $\Theta_1 = 21$ and $\Theta_2 = 120$ throughout the paper, but our contributions do not hinge on the specific choices of $\Theta_1$ and $\Theta_2$.

If $q$ is a query and $d$ a document (web page), we say that $q$ *covers* $d$ in case $d$ is of rank lower than $\Theta_1$ in the result list returned by the search engine upon query $q$. By a *low (high) rank* we refer to a low (high) rank number, *i.e.*, documents with a low rank occur high up in the result list.

We assume a user interacting with the search engine in order to satisfy a specific information need. The user's first query $q_0$ in such an interaction is called the *initial query*. The *original rank* $r_d$ of a document $d$ is its rank in the result list returned upon the initial query (the minimal such rank number, in case $d$ occurs several times in the result list).

## II. Underlying assumptions and objectives

Our system-centered evaluation rests on assumptions with implications on the objectives according to which we implement and evaluate our system.

***Models of rank vs. relevance:*** A first crucial assumption our approach builds on is that the initial query posed by the user is good enough for relevant documents to occur "high enough" in the result list (up to rank $\Theta_2$). Moreover, our system is designed in a way that the set of suggested queries maximizes the probability that the one the user will pick covers some document(s) relevant to the user. For this purpose we need not only appropriate values for the thresholds $\Theta_1$ and $\Theta_2$, but also a likelihood function that expresses, for every rank $r$, the probability that the document originally ranked $r$ is relevant to the user.

Two such likelihood functions, based on two different assumptions, are given in [5]. Assumption *LastRel* presumes that the user stops searching as soon as the first relevant document is found. Assumption *AllRel* presumes that the user may click several relevant documents before stopping. Either of these likelihood functions could be used in our system's objective functions and for evaluation. However, they almost certainly substantially underestimate the number of documents with ranks below 300, because they assume that relevant documents that did not appear in the first 300 results still exist on the web but had ranks greater than 300. In reality, because the log analyzed to create the likelihood function is several years old, a substantial number of these relevant documents are no longer on the web at all, so their rank today is simply undefined, not greater than 300. We therefore conducted a simple experiment in order to update the probability values reported in [5]. Out of about 750,000 clicked URLs occurring in the transaction log dataset used in [5], we randomly sampled 50,000 clicked URLs (for *AllRel*) and another 50,000 URLs that were clicked just before users stopped searching (for *LastRel*), both only out of the cases where the clicked URL was not ranked in the top 300 in the user's initial query. We tested what percentage of these URLs no longer exist on the web and multiplied the percentage by the percentage of URLs that are not ranked in the top 300 among all the clicked URLs. Thus we found that 13.6% of all clicked URLs are missing in the *AllRel* case and that 11.8% of all clicked URLs are missing in the *LastRel* case. Consequently, we updated every probability value $p$ of a document being relevant given its rank by adding $p \cdot x/(100 - x)$ for $x = 13.6$ and $x = 11.8$ respectively.[1] We will use

$$P_{\mathrm{pr}}(d \text{ relevant} \mid \mathrm{rank}(d) = r_d)$$

to denote the likelihood function resulting from this update of the values given in [5], where this notation represents the probability that a document $d$ is relevant given its original rank $r_d$. Whether it is for *LastRel* or *AllRel* will be indicated explicitly in the accompanying text.

As an alternative to using the updated likelihood functions from [5] to give "weights" to documents, we also consider a model that gives equal weight to the documents with an original rank in $[\Theta_1, \Theta_2]$. We do this by using a likelihood function according to which all these documents have the same probability of being relevant, *i.e.*, for some constant $c > 0$

$$P_{\mathrm{c}}(d \text{ relevant} \mid \mathrm{rank}(d) = r_d) = \begin{cases} c\,, & \text{if } \Theta_1 \le r_d \le \Theta_2\,, \\ 0\,, & \text{otherwise} \end{cases}$$

***User models:*** When measuring the system performance we would like to determine the probability that a user is satisfied with the query suggestions. If only one of the queries suggested yields satisfactory results for the user then it is crucial whether or not we believe the user will recognize this query suggestion among the "useless" ones. Let $Q^* = \{q_1, \ldots, q_m\}$ be the set of query suggestions and $D^*$ the set of documents covered by at least one $q_i$. For every $d \in D^*$, $r_d$ denotes the original rank of $d$. Let $z$ be in $\{\mathrm{pr}, \mathrm{c}\}$. Two extreme cases can be described as follows.

(1) The user always recognizes a satisfactory query suggestion. In this case, the objective of the query suggestion system is to maximize the probability that at least one of the $q_i \in Q^*$ covers some relevant document(s), which is

$$\lambda_{\mathrm{opt},z}(Q^*) = 1 - \prod_{d \in D^*} \left(1 - P_z(d \text{ relevant} \mid \mathrm{rank}(d) = r_d)\right).$$

(2) The user just picks a query suggestion at random. In this case, the objective of the query suggestion system is to maximize

[1]Note that this is optimistic since a missing URL does not imply the document previously addressed by that URL is no longer on the web itself. This introduces some additional noise to the likelihood functions we use, but probably gives a more realistic picture than assuming all these URLs are just ranked beyond 300 in their respective user sessions.

$$\lambda_{\text{rnd},z}(Q^*) = \frac{1}{m}\sum_{i=1}^{m}[1 - \prod_{d \in D_i}(1 - P_z(d \text{ relevant} \mid \text{rank}(d) = r_d))],$$

where $D_i$ is the set of documents covered by $q_i$.

Note that these cases are extremes; the truth probably lies in between those two cases.

Moreover, for evaluation purposes, if a document satisfying the user's information need is already covered by the initial query then according to our assumptions the user does not need any query suggestions. Therefore, when measuring the performance of our system, we assume that the documents covered by the initial query are not relevant. Consequently, when using $P_{\text{pr}}$, the probability that a document originally ranked in $[\Theta_1, \Theta_2]$ is relevant increases. To account for that, we divided the corresponding probability values by $1 - \sum_{j \in \{1,\ldots,20\}} P_{\text{pr}}(d \text{ relevant} \mid \text{rank}(d) = j)$ whenever $P_{\text{pr}}$ was used. Furthermore, whenever $P_{\text{c}}$ was used, given the constant value $c$, $\lambda_{\text{opt,c}}$ and $\lambda_{\text{rnd,c}}$ only depend on the number of documents in $D^*$ (or $D_i$) that are originally ranked in $[\Theta_1, \Theta_2]$. In what follows, these numbers are reported instead of $\lambda_{\text{opt,c}}$ and $\lambda_{\text{rnd,c}}$. We denote the changed measures using a capital $\Lambda$ rather than $\lambda$. This yields four measures, $\Lambda_{\text{opt,pr}}$, $\Lambda_{\text{opt,c}}$, $\Lambda_{\text{rnd,pr}}$, $\Lambda_{\text{rnd,c}}$, that can be taken as objective functions in our system and for the evaluation of our system; $\Lambda_{\text{opt,pr}}$ and $\Lambda_{\text{rnd,pr}}$ can use either assumption *LastRel* or assumption *AllRel*. Note that, under assumption *AllRel*, the value for $P_{\text{pr}}$ depends on the expected number of relevant documents for a given query. We set this number to 5 when $P_{\text{pr}}$ was used in our system's objective function and used three different values for it (1, 5, and 10) in evaluating our system.

## III. Query suggestion system

The process by which our system determines query suggestions for an initial query $q_0$, consists of these steps:

1) Issue the query $q_0$ to the Google search engine. The documents ranked in the range $[\Theta_1, \Theta_2]$ in the result list form the set $D$ of *reference documents*.
2) For every $d \in D$, construct a set $Q_d$ of queries such that every query in $Q_d$ covers $d$. $Q^c = \bigcup_{d \in D} Q_d$ is the set of all *candidate queries*.
3) Select $m$ queries from $Q^c$ so as to maximize a given objective function. The selected queries are returned as query suggestions.

### A. Construction of candidate queries

In the query construction process first for every reference document a single *root query* is constructed. This root query has the property that it covers the reference document and consists of a bounded number $t$ of terms, $t \in [T_1, T_2]$ (in our implementation $T_1 = 5$ and $T_2 = 10$; a term for us is any string that does not contain a space symbol).

The root query for a document $d$ is constructed as follows. First, a *seed sequence* is built by removing stop words and non-title terms with a term frequency less than 3 from $d$, then ordering the remaining terms. The seed sequence starts with the title terms of $d$ (in the same order as in the title) followed by the other terms in order of decreasing term frequency. The root query $q_d$ is the shortest initial segment of the resulting seed sequence $S$ that has at least $T_1$ and at most $T_2$ terms and that covers $d$. The latter property is tested by issuing the query to the Google API (*University Research Program for Google Search*). In case no such initial segment exists, the same procedure is repeated with a modified seed sequence $S'$ created by prepending the initial query to $S$. This slightly increases the success with which root queries are found.

For every reference document $d$ for which a root query is found, a set of candidate queries is derived. The candidate queries for $d$ still cover $d$ but they are simpler than the root query for $d$ and thus usually cover not only $d$ but also some other reference documents, so that among the set of candidate queries for all reference documents we hope to be able to find a relatively small number of query suggestions that cover a large number of reference documents. The construction of candidate queries involves a *query search* technique as proposed in [9]. In a top-down search over the space of all subqueries of the root query we search for the set of shortest subqueries that cover $d$. To prune the search tree, we do not shorten a subquery any further if it does not cover $d$, assuming that shorter queries are less likely to bring $d$ to the top of the result list. Moreover, we bound the length of query candidates to be in an interval $[V_1, V_2]$.[2] In pseudocode, this looks as follows.

```
for all $d \in D$ and corresponding root query $q_d$ do
  $C_d := [q_d]$;
for all $d \in D$ do
  $Q_d := ()$;
  while $C_d$ is not empty do
    $q :=$ any element in $C_d$;
    remove $q$ from $C_d$;
    if $length(q) \in [V_1, V_2]$ then add $q$ to $Q_d$;
    for all $q^-$ created by deleting a term from $q$ do
      if $q^-$ covers $d$ and $length(q) \geq V_1$ then
        add $q^-$ to $C_d$;
        if $q \in Q_d$ then remove it ;
        if $length(q^-) \leq V_2$ then
          for all $d' \in D$ covered by $q^-$ do
            add $q^-$ to $C_{d'}$;
```

Note that reference documents $d'$ without root queries can have candidate queries if they are covered by queries constructed from the root queries of other reference documents.

### B. Selection of query suggestions

After the construction of candidate queries most of the relevant documents are covered by at least one candidate

[2]In our experiments we used $V_1 = 2$ and $V_2 = 5$.

query (we do not have a theoretical guarantee for this, but it is shown empirically by the results in Section IV-B and in [9]). However, in order to keep the list of query suggestions short enough for the user to see them all in a single glance, we have to select a subset of $m$ queries (we used $m = 10$ in our implementation). We do this by greedily selecting queries from the set $Q^c$ of query candidates to maximize either $\Lambda_{\mathrm{opt,pr}}$ (for *LastRel* or *AllRel*) or $\Lambda_{\mathrm{opt,c}}$.

This simple greedy selection turned out to yield results slightly inferior to those obtained by a variant involving a one-step lookahead greedy selection. In particular, this variant always selects a query for which there remains a second query candidate such that the value of the objective function for that pair of queries achieves the highest increase.

All results given below are based on this lookahead variant. Regarding speed, obviously the greedy query selection procedure is very quick. The construction of candidate queries requires only a small number of queries to be issued on average (empirically about 30). If run "inside Google" this would take a fraction of a second. Alternatively, the construction of candidate queries can be done ahead of time, for example when a document is indexed.

## IV. System evaluation

To evaluate our system, we sampled 250 queries consisting of at most 2 terms (*short queries*) and 250 queries consisting of at least 3 terms (*long queries*), where stop words were not counted as terms; all queries were randomly sampled from an AOL transaction log recorded in 2006[3]. The two groups of 250 user queries formed separate test sets. For each test set we ran and evaluated our implementation in different variants according to the different criteria introduced in Section II.

### A. Parameter tuning

To obtain the results described here we tuned a series of parameters. These included the rank threshold used in the definition of "covering" when constructing root queries, different algorithmic variants of dealing with the case that no root query is found if the initial query is not included, the definition of "term", whether or not phrases were allowed in the construction of root queries, and different algorithmic variants for the query selection. A detailed discussion of all the variants tested is beyond the scope of this paper. What is described above and evaluated here is the most successful version resulting from tuning the parameters one after the other in the given order and always fixing a parameter to its tuned value before the next parameter was tuned. The reader is referred to [4] for more details.

[3]This log was downloaded from http://gregsadetsky.com/aol-data/ and used in [10].

### B. Results and interpretation

98.40% of the short queries (97.20% of the long queries) had a non-empty set of reference documents, *i.e.*, had more than 20 documents in their result list when issued to Google. On average, 93.09% (90.72% for long queries) of the corresponding reference documents were still existing html web pages (our system does not support other documents (*e.g.* pdf files) so far). Over those, for on average 85.59% (87.25% for long queries) the construction of a root query was successful. The average length of the constructed root queries was 5.22 terms (5.39 terms for long queries). Finally, on average 91.23% (87.70% for long queries) of all reference documents were assigned at least one query suggestion candidate.

Our system achieved performance values as shown in Table I for short queries and Table II for long queries.

| Assumption | AllRel | AllRel | LastRel | LastRel |
|---|---|---|---|---|
| Criteria | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,c}}$ | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,c}}$ |
| $\Lambda_{\mathrm{opt,pr}}[1]$ | 11.08% | 10.01% | **11.22%** | 9.78% |
| $\Lambda_{\mathrm{opt,pr}}[5]$ | 44.23% | 40.67% | **44.72%** | 39.97% |
| $\Lambda_{\mathrm{opt,pr}}[10]$ | 68.55% | 64.26% | **69.18%** | 63.49% |
| $\Lambda_{\mathrm{rnd,pr}}[1]$ | 1.56% | 1.40% | 1.59% | 1.38% |
| $\Lambda_{\mathrm{rnd,pr}}[5]$ | 7.54% | 6.76% | **7.65%** | 6.67% |
| $\Lambda_{\mathrm{rnd,pr}}[10]$ | 14.44% | 12.98% | **14.64%** | 12.81% |
| $\Lambda_{\mathrm{opt,c}}$ | 46.7 | **52.3** | 45.1 | **52.3** |
| $\Lambda_{\mathrm{rnd,c}}$ | 6.0 | **6.5** | 5.8 | **6.5** |

Table I
EXPERIMENTAL RESULTS (SHORT QUERIES)

| Assumption | AllRel | AllRel | LastRel | LastRel |
|---|---|---|---|---|
| Criteria | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,c}}$ | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,c}}$ |
| $\Lambda_{\mathrm{opt,pr}}[1]$ | 10.57% | 9.44% | **10.76%** | 9.28% |
| $\Lambda_{\mathrm{opt,pr}}[5]$ | 42.63% | 38.76% | **43.30%** | 38.24% |
| $\Lambda_{\mathrm{opt,pr}}[10]$ | 66.83% | 61.93% | **67.66%** | 61.32% |
| $\Lambda_{\mathrm{rnd,pr}}[1]$ | 1.47% | 1.30% | 1.50% | 1.29% |
| $\Lambda_{\mathrm{rnd,pr}}[5]$ | 7.09% | 6.30% | **7.25%** | 6.24% |
| $\Lambda_{\mathrm{rnd,pr}}[10]$ | 13.61% | 12.10% | **13.91%** | 12.00% |
| $\Lambda_{\mathrm{opt,c}}$ | 42.5 | **47.6** | 40.9 | **47.6** |
| $\Lambda_{\mathrm{rnd,c}}$ | 5.4 | **5.9** | 5.2 | **5.9** |

Table II
EXPERIMENTAL RESULTS (LONG QUERIES)

The columns represent the criteria used in our objective functions; the rows represent those used for evaluation. The numbers in square brackets denote the number of relevant documents for a given query that are assumed to exist in the corpus. Numbers in boldface are those explained below.

Each number in the first 6 rows is the probability that the query suggestion the user selects will cover at least one relevant document. In the short query case for instance, if $\Lambda_{\mathrm{opt,pr}}$ is chosen as an objective function, $P_{\mathrm{pr}}$ is derived from the *LastRel* assumption, the system implementation assumes there are 5 relevant documents but in fact there are 10 relevant documents, then the user has a chance of 69.18%

to find a relevant document satisfying his/her information need in case he/she immediately recognizes a useful query among the 10 query suggestions made by our system (or looks at them all). If there are in fact only 5 relevant documents, the chance decreases to 44.72%. Note that if a user looks at only one query suggestion at random, the chances of being successful are only 14.64% and 7.65%, but this is an extreme case that we consider unlikely. All these numbers are measured under the assumption that the user was unhappy with the top 20 documents returned by the initial query.

The results when there is just one relevant document and the user chooses a satisfactory suggestion seem low (11.22% for short queries, 10.76% for long queries), but note that the case that only one relevant document exists for a query is relatively unlikely (unless a URL is issued as a query—a case that is filtered out in all our experiments and the experiments in [5]) and obviously expected to be a tough case for any query suggestion system.

Each number in the bottommost two rows is an average absolute number of reference documents. In the short query case, there are on average 52.3 reference documents covered collectively by the 10 query suggestions, with each individual query covering 6.5 reference documents on average (note that some relevant documents are covered by more than one suggested query). If $P_{\mathrm{c}}$ is used both in the objective function and for evaluation (second and fourth column of numbers, bottommost two rows) then of course assumptions *AllRel/LastRel* have no influence.

In general, the results show that our approach to query suggestion is slightly more successful if the initial query is short. This is quite intuitive since short queries are less specific than long queries and thus a small set of refined queries can more easily cover a large set of reference documents by clustering them into subtopics. We also observe that the combination of assumption *LastRel* with $\Lambda_{\mathrm{opt,pr}}[5]$ yields the overall best results when evaluated with *any* measure relying on $P_{\mathrm{pr}}$.

The numbers shown in Tables I and II should not be seen as true optimal values achievable by our approach. Several facts indicate that *(i)* these numbers are pessimistic estimates and the true performance values of the implemented system are higher, and *(ii)* further system improvements resulting in provably higher performance values are possible.

*Regarding (i).* Our query suggestions cover not only documents with an original rank in $[21, 120]$ but also some with a higher original rank. Those have a certain probability of being relevant that is not accounted for in our evaluation (we only account for documents in the range $[21, 120]$). In fact, one can argue that if a query $q'$ covers several relevant documents with an original rank in $[21, 120]$ plus some documents with a higher original rank, then the latter documents have a much higher chance of being relevant than their expected relevance value. The reason is that they are probably semantically related to the relevant documents that are covered by $q'$.

*Regarding (ii).* Evaluating our system, we did not take into account potential changes to the user interface used in web search. In our evaluation we do not count the documents covered by the suggested query $q'$ if they were already covered by the initial query $q_0$–since we consider them as already being viewed by the user initially. In that sense, those documents use up space in the result list for $q'$ unnecessarily. If the user selects a query suggestion the interface could display in the results only those documents that were not already covered by the initial query $q_0$. Consequently, the new top 20 results for $q'$ could contain more documents with an original rank in $[21, 120]$. We evaluated our system a second time using this approach and achieved substantially higher values, see the columns labeled "new" in Table III, compared to the previous values (columns labeled "old"), for both short and long queries.

|  | old (short) | new (short) | old (long) | new (long) |
|---|---|---|---|---|
| Criteria | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,pr}}[5]$ | $\Lambda_{\mathrm{opt,pr}}[5]$ |
| $\Lambda_{\mathrm{opt,pr}}[5]$ | 44.72% | **46.67%** | 43.30% | **45.77%** |
| $\Lambda_{\mathrm{opt,pr}}[10]$ | 69.18% | **71.30%** | 67.66% | **70.41%** |
| $\Lambda_{\mathrm{rnd,pr}}[5]$ | 7.65% | **8.91%** | 7.25% | **8.79%** |
| $\Lambda_{\mathrm{rnd,pr}}[10]$ | 14.64% | **16.81%** | 13.91% | **16.55%** |
|  | $\Lambda_{\mathrm{opt,c}}$ | $\Lambda_{\mathrm{opt,c}}$ | $\Lambda_{\mathrm{opt,c}}$ | $\Lambda_{\mathrm{opt,c}}$ |
| $\Lambda_{\mathrm{opt,c}}$ | 52.3 | **57.0** | 47.7 | **53.1** |
| $\Lambda_{\mathrm{rnd,c}}$ | 6.5 | **7.7** | 5.9 | **7.3** |

Table III
EXPERIMENTAL RESULTS FOR SHORT AND LONG QUERIES UNDER ASSUMPTION *LastRel* AND A CHANGED INTERFACE

## V. RELATED WORK

One technique strongly related to query suggestion is *query expansion*, where queries are refined by adding terms to them, see, *e.g.*, [11]. Other methods of changing the initial query to a suggested query, are based on, for example, substituting one or more query terms [6], [13]. Our method is novel in the way candidate terms or queries are obtained. Unlike most existing systems, they are not selected based on the similarity to the initial query terms or queries. Instead, we employ the query search technique to construct query suggestion candidates.

[12] uses a kind of queries similar to our candidate queries for document summarization. Such queries, called *query associations*, are used for query expansion in [1], but there the candidates stem from query associations of documents with a low original rank. Query associations are taken from a transaction log, while we extract them from reference documents by query search. As discussed in [2], the terms used by users and the terms in a corpus are usually very different.

Besides query suggestion and query expansion, there are other techniques for user assistance in web search, such as *Result Clustering*, which has been adopted by many popular search engines, *e.g.*, Vivisimo [7]. Here the retrieval results with an original rank in a certain range are clustered and each cluster is assigned a label. The cluster labels are listed in the result page beside the normal retrieval results. Query suggestion can be viewed as clustering with a special labeling; each suggested query can be interpreted as a cluster label, and the documents covered by each suggested query form a cluster. One important difference between query suggestion and clustering is that query suggestion does not only cluster documents within a certain original rank range but also other documents in the corpus due to their being covered by a suggested query. See, e.g., [3], [8], [14] for more on result clustering.

## VI. Conclusions

We introduced and analyzed a new approach to query suggestion and evaluation of query suggestion systems.

The construction of query suggestion candidates relies purely on the documents in a specific original rank range and uses a query search technique [9]. The objective functions we proposed are of general value and can be used for evaluation of other query suggestion systems. This results in a new approach to system-centered evaluation of query suggestion.

The evaluation of our system validates our approach and shows clear tendencies as to which objective functions are favourable in our framework. The suggestions made by our system support the intuitive idea that query expansion in general is a good strategy for query suggestion but not quite sufficient. Many suggested queries, especially if the initial query is short, are actually query expansions, but some are not (note though that most contain all but one original term). This is remarkable since, unlike most systems, our system does not intentionally construct queries to have such properties except in those rare cases when the primary method of constructing seed queries fails. These properties here are shown to be favourable at least when using the Google search engine.

## Acknowledgments

This research draws on data provided by the University Research Program for Google Search, a service provided by Google to promote a greater common understanding of the web. We also gratefully acknowledge support by the Alberta Ingenuity Centre for Machine Learning and thank Shane Bergsma, Christopher Pinchak (both of the University of Alberta, Edmonton), and Joel Martin (National Research Council, Ottawa) for their assistance in early stages of this work.

## References

[1] B. Billerbeck, F. Scholer, H.E. Williams, and J. Zobel. Query expansion using associated queries. In *Proc. 12th International Conf. on Information and Knowledge Management*, pages 2–9. ACM, 2003.

[2] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proc. 11th International Conf. on World Wide Web*, pages 325–332. ACM, 2002.

[3] M.A. Hearst and J.O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proc. 19th ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 76–84. ACM, 1996.

[4] S. Jiang. *Searching for Queries to Improve Document Retrieval in Web Search.* M.Sc. thesis, Department of Computing Science, University of Alberta, 2009.

[5] S. Jiang, S. Zilles, and R. Holte. Empirical analysis of the rank distribution of relevant documents in web search. In *Proc. IEEE/WIC/ACM International Conf. on Web Intelligence*, pages 208–213, 2008.

[6] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. 15th International Conf. on World Wide Web*, pages 387–396. ACM, 2006.

[7] S. Koshman, A. Spink, and B.J. Jansen. Web searching on the Vivisimo search engine. *JASIST*, 57(14):1875–1887, 2006.

[8] A.V. Leouski and W.B. Croft. An evaluation of techniques for clustering search results. Technical report, Dept. of Computer Science, University of Massachusetts, Amherst, 1996.

[9] J.D. Martin and R.C. Holte. Searching for content-based addresses on the world-wide web. In *Proc. 3rd ACM Conf. on Digital Libraries*, pages 299–300. ACM, 1998.

[10] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proc. 1st International Conference on Scalable Information Systems*, page 1. ACM, 2006.

[11] I. Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *Proc. 26th ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 213–220. ACM, 2003.

[12] F. Scholer and H.E. Williams. Query association for effective retrieval. In *Proc. 11th International Conf. on Information and Knowledge Management*, pages 324–331. ACM, 2002.

[13] N. Stojanovic. Information-need driven query refinement. *Web Intelligence and Agent Systems*, 3(3):155–169, 2005.

[14] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. *Computer Networks*, 31(11-16):1361–1374, 1999.