World Wide Web (2014) 17:1141–1160
DOI 10.1007/s11280-013-0235-3

Assisting web search using query suggestion based on word similarity measure and query modification patterns

Rani Qumsiyeh · Yiu-Kai Ng

Received: 28 August 2012 / Revised: 4 June 2013 /
Accepted: 14 June 2013 / Published online: 13 July 2013

Abstract One of the useful tools offered by existing web search engines is *query suggestion* (QS), which assists users in formulating keyword queries by suggesting keywords that are unfamiliar to users, offering alternative queries that deviate from the original ones, and even correcting spelling errors. The design goal of QS is to enrich the web search experience of users and avoid the frustrating process of choosing controlled keywords to specify their special information needs, which releases their burden on creating web queries. Unfortunately, the algorithms or design methodologies of the QS module developed by Google, the most popular web search engine these days, is not made publicly available, which means that they cannot be duplicated by software developers to build the tool for specifically-design software systems for enterprise search, desktop search, or vertical search, to name a few. Keyword suggested by Yahoo! and Bing, another two well-known web search engines, however, are mostly popular currently-searched words, which might not meet the specific information needs of the users. These problems can be solved by WebQS, our proposed web QS approach, which provides the same mechanism offered by Google, Yahoo!, and Bing to support users in formulating keyword queries that improve the precision and recall of search results. WebQS relies on *frequency of occurrence*, *keyword similarity measures*, and *modification patterns* of queries in user query logs, which capture information on millions of searches conducted by millions of users, to suggest useful queries/query keywords during the user query construction process and achieve the design goal of QS. Experimental results show that WebQS performs as well as Yahoo! and Bing in terms of effectiveness and efficiency and is comparable to Google in terms of query suggestion time.

Keywords Query suggestion · Word similarity · Query patterns · Query frequency

R. Qumsiyeh · Y.-K. Ng (✉)
Computer Science Department, Brigham Young University, Provo, UT 84602, USA
e-mail: ng@compsci.byu.edu

R. Qumsiyeh
e-mail: raniq@microsoft.com

1 Introduction

Web search engine users often provide *imprecise* specifications of their information needs in the form of keyword queries, either because they are in a hurry, use inappropriate keywords, or do not understand the search process well. These scenarios might explain why web search engine users frequently create short queries,[1] which are *incomplete* or *ambiguous*. Providing a user interface to assist users in constructing keyword queries that capture their information needs is an essential design issue of web search, which can significantly enhance the precision of search results [29].

Current web search engines, such as Google, Yahoo!, and Bing,[2] are equipped with a *query suggestion* (QS) module that provides a guide to its users in formulating keyword queries, which facilitates the web search. Any web search conducted by a user U is supported by the QS module of these popular search engines through a query-creation interface which suggests potential keywords to be included in a query being constructed by U.

Keywords suggested by current web search engines, however, are mostly popular currently-searched keywords, which might not be the most relevant keywords that specify the information requested by the users. For example, when the word "Tiger" is entered by a user, current web search engines are mostly focused on the query "Tiger Woods", a golf player, instead of queries related to any animal, airlines, or others named "Tiger". While the suggested query keywords are good for the users who look for information on "Tiger Woods", it is not so for users who are interested in a sport team or an airlines associated with the keyword "Tiger". This drawback can be overcome by considering the past, besides current, general *search patterns* of web search engine users in query logs. Furthermore, when considering query keywords in query logs that match a user's entered keywords, existing web search engines consider exactly the same ones, but exclude the ones *similar in content* that might capture the users' information needs. For example, "car" and "vehicle" are similar and often interchangeably used, which should be treated as potential keywords to be suggested simultaneously to a user who enters either the keyword "car" or "vehicle". Considering closely-related keywords for query suggestions should enhance the precision of a web search.

To enhance and deal with the shortcomings of the QS module of existing web search engines, we introduce a web query suggestion approach, denoted WebQS, which provides a guide to the users for formulating/completing a keyword query Q using suggested keywords (extracted from query logs) as potential keywords in Q. WebQS considers initial and modified queries in query logs, along with word-similarity measures in making query suggestions. The proposed WebQS facilitates the formulation of queries in a *trie* data structure and determines the rankings of suggested keyword queries using distinguished features exhibited in the raw data in query logs. The design of WebQS is *simply* and *unique*, which does not require

[1]The AOL query logs show that 84 % of submitted queries are either unigram or bigram queries.

[2]It appears that the alliance agreement between Yahoo! and Bing does not cover the query suggestion module, since manual examination indicated that queries suggested by the two search engines were different.

any training data, machine learning approaches, knowledge bases, nor ontologies to suggest appropriate keywords to the users for constructing queries.

Along with Google, Yahoo!, and Bing, WebQS saves its users' time and effort in formulating a search query, since WebQS suggests keywords to be used before a user has entered the entire query. Besides detecting spelling errors in queries being created, WebQS suggests specific queries for the suffix strings of potential queries extracted from query logs.

We have compared WebQS with Google, Yahoo!, and Bing in terms of the time required to formulate user queries using the corresponding QS module. In addition, we have conducted several controlled experiments to analyze the user satisfaction of WebQS in suggesting and ranking search queries. The performance evaluation validates the effectiveness and efficiency of WebQS and compares its performance with the QS models of Google, Yahoo!, and Bing. Experimental results show that WebQS is highly efficient in suggestion query keywords, which is comparable in performance with Google, Yahoo!, and Bing. The empirical study also verifies that in general queries suggested by WebQS are ranked as high as the ones recommended by Yahoo! and Bing, respectively.

The remaining sections in this paper are organized as follows. In Section 2, we discuss works related to query suggestion. In Section 3, we introduce the query suggestion, spelling correction, and ranking modules of WebQS. In Section 4, we present the experimental results which assess the performance of WebQS in terms of its effectiveness and efficiency and compare its performance with Google, Yahoo!, and Bing. In Section 5, we give a conclusion.

2 Related work

Recent developments on QS combine the co-occurrence of keywords and a hand-crafted thesaurus. Cao et al. [4] propose a language model that captures word relationships by utilizing WordNet, a hand-crafted thesaurus, and word co-occurrence computed using co-occurrence samples. Even though Liu et al. [19] achieve an improved performance on query suggestion using WordNet, words and their measures in WordNet are subjective and, unlike user query logs, do not capture (i) relationships among keywords from the users' perspective and (ii) updated keyword relationships through time. Ruch et al. [26] present an argumentative feedback strategy in which suggested query terms are selected from sentences classified into one of the four disjunct argumentative categories that have been observed in scientific reports. Since the feedback strategy tailors for scientific reports, it cannot cover the heterogeneous Web.

Cao et al. [5] introduce a query suggestion approach based on the contexts of queries recently issued by a user. Context-based query suggestion, however, requires very large query logs, since keywords suggested for a user query must appear in a list of related queries varying in size. Boldi et al. [3] utilize the *Query Flow Graph* (QFG), a directed graph in which nodes are queries and an labeled edge from node q_i to node q_j indicates the probability of q_j being a suggestion for q_i. QFG is utilized for creating suggested queries based on a random walk with a restart model. Baraglia et al. [1] modify QFG to allow graphs to be updated to enhance their efficiency. The log-based methods in [1, 3] are adequate for frequently-created queries, since accurate statistics exist. The statistics for infrequent queries, however, are based on a few instances, which can lead to poor suggestions. In addition, the two log-based QS methods rely on *training* to compute edge weights, which is not required by WebQS.

Liao et al. [18] propose a context-aware query suggestion method which considers the immediately preceding queries in query logs as context to suggest queries. The authors first construct a concept sequence suffix tree which captures queries summarized into concepts and then during the query suggestion process, a user's search context is mapped to a sequence of concepts. This QS approach, however, relies on the accuracy of concept analysis, which imposed an additional design issue that is not employed by WebQS. Kato et al. [16] construct a prototype, called SParQS, which classifies query suggestions into labeled categories to handle query reformulation–specialization and parallel movement. The prototype requires a classification process, which is another overhead component of the query suggestion process.

Without using query logs, Bhatia et al. [2] develop a probabilistic mechanism for generating query suggestions which extracts candidate phrases of suggested queries from document corpus. The mechanism, however, cannot be generalized, since it applies only to customized search engines for enterprise, intranet, and personalized searches. Song and He [27] present an optimal rare query suggestion framework, which infers implicit feedback from users in query logs. The framework is based on the commonly-used pseudo-relevance feedback strategy which assumes top-ranked results by search engines are relevant. Since queries considered by the authors of [27] are rare queries, its approach is not applicable to general queries created by common users. The same problem applied to the query suggestion model proposed by the authors of [12] which relies on the context of e-commerce marketplace.

Widely-used web search engines, such as Google, Yahoo!, and Bing, assist users with query suggestion. We have observed that (i) Google has the fastest QS module and (ii) keywords suggested by existing web search engines for a user query do not seem to differ significantly from one to the other. Furthermore, the QS modules of these web search engines are based on either the *popularity* (as mentioned in the Introduction section) or *morphological information* of queries, i.e., co-occurrence of a query word with other query words [21]. Although such QS modules are useful in guiding the user through the process of constructing a query, occasionally, the suggested queries may not match the semantic of the query that the user intends to create. For example, an intended web search for "Harry Shum" would yield the suggested query "Harry Potter" after the first keyword has been entered, although the two are not related. Besides considering frequency of co-occurrence, WebQS suggests query keywords based on *word similarity* and *modified* queries.

3 WebQS

Query suggestion (QS) can be categorized into automatic and interactive. *Interactive QS* displays recommended keywords for a query being created by a user who can choose one of the suggested queries or submit his own to the corresponding search engine. *Automatic QS*, on the other hand, processes a user query Q without providing suggested keywords to the user while the user is entering a query. Instead, Q is expanded internally using related keywords before it is processed by the corresponding search engine. Both approaches require a "query log" and a mechanism for deriving and ranking the suggested/expanded keywords. WebQS is an interactive QS module, which utilizes the AOL query logs (discussed in Section 3.1), incorporates a trie structure for deriving suggested query words (presented in Section 3.2), detects spelling errors in queries being constructed (detailed in Section 3.3), and applies a feature-based algorithm for ranking suggested queries (see Section 3.4).

3.1 The AOL query logs

WebQS relies on the AOL query logs to suggest queries. The logs of AOL (gregsadetsky.com/aol-data/), which include 50 million queries (among which about 30 million are unique and used by WebQS) that were created by millions of AOL users over a three-month period between March 1, 2006 and May 31, 2006. The query logs are publicly available.

An AOL query log includes a number of query sessions, each of which captures a period of sustained user activities on the search engine. Each AOL session differs in length and includes a (i) user ID, (ii) the query text, (iii) date and time of search, and (iv) optionally clicked documents. (Figure 1 shows the snapshot of a query session extracted from the AOL query logs.) A *user ID*, which is an anonymous identifier of its user who performs the search, determines the boundary of each session (as each user ID is associated with a distinct session). *Query text* are keywords in a user query and multiple queries may be created under the same session. The *date* and *time* of a search can be used to determine whether two or more queries were created by the same user within 10 min, which is the time period that dictates whether two queries should be treated as *related* [9]. *Clicked documents* are retrieved documents that the user has clicked on and are ranked by the search engine. Queries and documents include stopwords, which are commonly-occurring keywords, such as prepositions, articles, and pronouns, that carry little meaning and often do not represent the content of a document. Stopwords are not considered by WebQS during the query creation process. From now on, unless stated otherwise, whenever we refer to "(key)words", we mean "non-stop (key)words".

Unique queries in AOL query logs are examined and suggested keywords are extracted automatically, whereas the duplicates are used as one of the features (as discussed in Section 3.4) to determine the ranking of a suggested query.

3.2 Processing the query logs

WebQS parses the AOL query logs to extract query keywords while at the same time retains the information of *related* keywords in the same session, which were submitted by the same user within 10 min in the same session, as discussed earlier. Using the extracted keywords, WebQS constructs a trie *T* in which each node is labeled by

User ID	Query	Date	Time	Clicked Documents
1326	back to the future	2006-04-01	17:59:28	http://www.imdb.com
1326	adr wheels	2006-03-28	12:53:39	
...				

Figure 1 An AOL query session

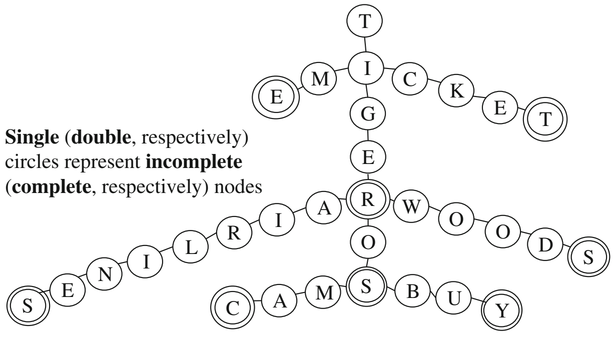
a letter in an extracted keyword in the given order, and each node in T is categorized as either "complete" or "incomplete". A *complete* node is the last node of a path in T representing an (a sequence of, respectively) extracted query keyword (keywords, respectively). If node c is a complete node, then T_c (the subtree of T rooted at a child node of c) contains other suggested keyword(s) represented by the nodes in the path(s) leading from, and excluding, c. The possible number of suggestions of a (sequence of) keyword(s) K rooted at T_c is n, where n is the number of complete nodes in subtrees rooted at T_c, and K is the (sequence of) keyword(s) extracted from the root of T_c. An *incomplete* node is the last node of a path P in T such that P does not yield a (sequence of) word(s). If c is an incomplete node, then all subsequent nodes of c up till the first complete node are potential suggestions of keywords represented by the nodes in the path leading from, and including, c.

WebQS retains the keywords in query texts in a *trie* data structure using queries in the AOL query logs, which is done once, and the constructed trie is 51 megabyte in size. Using the trie, candidate keywords suggested for a query can be found and ranked dynamically. To suggest potential query keywords, WebQS locates a trie branch b up till the (letters in the) keywords that have been entered during the query creation process and extracts the subtrees rooted at the child nodes of the last node of b. The extracted suggestions are ranked using a set of features (presented in Section 3.4.2).

Example 1 Figure 2 shows a trie of sample queries in a query log. If a user enters the letters "TI", all branches rooted at the child nodes of node "I" up till the first *complete node* in each branch are retrieved, which include "Time". "Tiger", and "Ticket", since node "I" is an incomplete node. If the user enters "TIG", then the keyword "Tiger" is displayed. If the user has entered "Tiger", the subtree rooted at node 'R', which is a complete node, is processed and the keywords "Airlines", "OS", and "Woods" are appended to "Tiger" and showed to the user as suggested queries.

Compared with other existing query suggestion and spelling correction approaches using either the trie data structure and/or query logs [7, 10, 13], WebQS is simpler and yet effective. A trie data structure is also used by [7], which stores all queries in a query log with their probabilities among all the queries in the log for online spelling correction. Using the *noisy channel transformation* model, the online spelling correction approach computes the probability of a user's intended query c

Figure 2 A sample trie


T
I
M
E
C
K
E
T
G
E
R
A
I
R
L
I
N
E
S
W
O
O
D
S
O
S
M
A
C
B
U
Y
Single (**double**, respectively) circles represent **incomplete** (**complete**, respectively) nodes

given the potentially misspelled input query q, which relies on the *prior probability* of c. WebQS avoids using any probability and transformation models to determine correct spellings and suggest queries, and its trie data structure is *simple* and more *efficient* compared with its counterpart in [7], since it contains only query keywords (without their probabilities of occurrence in a query log) and can suggest intended queries without using a trained Markov n-gram transformation model (as in [7]) for matching intended queries and (misspelled) input queries, which imposes additional overhead.

Instead of relying on web search engines to suggest query keywords, the authors of [13] analyze the human side of query reformulation to assist users in refining queries. The analysis involves with studies through (i) click data in a query log, (ii) user's behavior in response to the initial set of results, i.e., the user feedback strategy, and (iii) detection of each type of query reformulations in the context of the AOL query logs. Using a constructed rule-based classifier and based on users' click behavior, query reformulations are automatically presented to the users. Even though the proposed query reformulation strategies based on AOL query logs offer a new approach for query suggestions, they require the creation of a taxonomy of query refinement strategies and the construction the rule-based classifier, which are dependent on the contents of different query logs and require significant preprocessing steps. Unlike [13], WebQS only relies on the query modification patterns extracted from a query log for query suggestions.

Exploring the use of massive web corpora and query logs, Gao et al. [10] propose a *ranker-based speller* which creates a list of candidate corrections for a given input query using the noisy channel model and applies various features to identify the likelihood of a candidate as a desired correction. The ranker is augmented with trained *web scale language models* (LMs) for query spelling correction and a *phrase-based error model* that determines the probability of the transformations among different multi-term phrases using a large number of query-correction pairs extracted from query logs. The ranker, LMs, and error model require probability analysis, training, and extraction of query-correct spelling pairs, respectively, which complicate the entire process of search query spelling correction.

3.3 Using a prefix trie to correct spelling mistakes in queries

Occasionally, a user *misspells* a word or forgets to add *spaces* in between keywords when posting a query Q. For example, a user might create the query, “Hondaaccord” or “honda accorr”. To enhance the *effectiveness* and *user-friendliness* of WebQS, spelling errors are detected and corrected automatically using the trie of words in online dictionary or dictionaries, which avoids returning non-relevant or no results to the user. Thus, besides suggesting query keywords, a different trie data structure is used by WebQS for spell checking,

During the process of parsing a user's query Q, WebQS scans through each (in)complete keyword K in Q by reading its letters one by one. If an end of a branch in the trie is encountered and no more characters are left in K, WebQS treats K as a *valid* keyword; otherwise, if there are more characters left in K, a *space* is inserted at the current position of K, assuming that the user has forgotten to add a space. However, if K is not recognized by the trie (i.e., none of the next letters in the trie matches the next letter in K), then K is treated as a *misspelled* word W.

WebQS compares W with the alternative keywords recognized by the trie, starting from the current node in the trie where W is encountered, using the "*similar_text*" function [22] which calculates their *similarity* based on the number of common characters and their corresponding positions in the strings. *Similar_text* returns the degree of similarity of two strings as a *percentage*. WebQS replaces the misspelled keyword in Q by the alternative one with the *highest* similarity percentage.

Example 2 Consider the user's query "notwierd". Using the prefix trie with the vocabulary extracted from an online dictionary (such as one of those shown in Section 3.4.1), WebQS scans through the characters in "notwierd" and adds a space between the characters 't' and 'w' in "notwierd", generating "not wierd". The second keyword now is "wierd". Since "wierd" is detected as a misspelled word, WebQS replaces it by "weird" using the prefix trie.

3.4 Ranking possible suggestions

WebQS ranks suggested query keywords in its trie data structure based on (i) the *frequency of occurrence* (*freq*) of the keywords in the AOL query logs, (ii) their *similarity* with the keywords submitted by a user based on the word-correlation factors (WCFs)[3] [17], and (iii) the *number of times* the keywords in user queries were *modified* (*Mod*) to the keywords in the suggested queries within 10 min as shown in the query logs.

3.4.1 Word correlation factors (WCFs)

The word-correlation factor between any two words i and j, denoted $WCF(i, j)$, were pre-computed using 880,000 documents in the Wikipedia collection (downloaded from http://www.wikipedia.org/)[4] based on their (i) *frequency of co-occurrence* and (ii) *relative distance* in each Wikipedia document as defined below.

$$WCF(i,j) = \frac{\sum_{w_i \in V(i)} \sum_{w_j \in V(j)} \frac{1}{d(w_i, w_j)+1}}{|V(i)| \times |V(j)|} \tag{1}$$

where $d(w_i, w_j)$ is the *distance*, i.e., the number of words in between any two words w_i and w_j in any Wikipedia document, $V(i)$ ($V(j)$, respectively) is the set of stem variations of i (j, respectively) in the Wikipedia collection, e.g., the stem variations of the word "computer" are "compute", "computation", "computational", "computing", etc., $|V(i)|$ ($|V(j)|$, respectively) is size of $V(i)$ ($V(j)$, respectively), and $|V(i)| \times |V(j)|$ is a *normalization* factor.

The Wikipedia collection is an ideal and unbiased choice for establishing word similarity, since (i) documents within the collection were written by close to 90,000 authors with different writing styles and word usage, (ii) the Wikipedia documents cover an extensive range of topics, and (iii) the words within the documents appear in

[3]This measure cannot be employed until at least one whole keyword is entered by the user.

[4]Words within the Wikipedia documents were *stemmed* (i.e., reduced to their root forms) and *stopwords* were removed.

a number of on-line dictionaries, such as 12dicts-4.0 (prdownloads.sourceforge.net/wordlist/12dicts-4.0.zip), Ispell (www.luziusschneider.com/SpellerHome.htm), and RandomDict (www.ime.usp.br/~yoshi/mac324/projeto/dicas/entras/words). Compared with *WCF*s, WordNet (http://wordnet.princeton.edu/) provides synonyms, hypernyms, holonyms, antonyms, etc. for a given word. There is, however, no partial degree of similarity measures, i.e., weights, assigned to any pair of words in WordNet. For this reason, word-correlation factors yield a more sophisticated measure of word similarity than WordNet.

Word correlation factors have been used for solving different content-similarity problems, which include plagiarism detection [23] and readability level analysis [24], to name a few. Furthermore, Strube and Ponzetto [28] use a Wikipedia collection for computing semantic relatedness of various concepts/words based on the *hierarchical structure* of the *Wikipedia category tree*. They have chosen Wikipedia for computing semantic relatedness, since its documents include a large number of name entities and are rich in concepts. The semantic relatedness of any two words in [28], however, is determined using paths and the hierarchical structure of the Wikipedia category tree, a data structure that is avoided by [17] in computing the correlation factors of two words, which is straightly based on co-occurrence and relative positions of words in Wikipedia documents.

3.4.2 Rankings using the stanford certainty factor

Given that SQ is a suggested query for a(n) (in)complete user query Q which has been entered during the query construction process, WebQS computes a *ranking* score for SQ, denoted $SuggRank(Q, SQ)$, which reflects the degree of *closeness* of SQ to the letters/keywords in Q.

$$SuggRank(Q, SQ) = \frac{freq(SQ) + WCF(Q, SQ) + Mod(Q, SQ)}{1 - Min\{freq(SQ), WCF(Q, SQ), Mod(Q, SQ)\}} \tag{2}$$

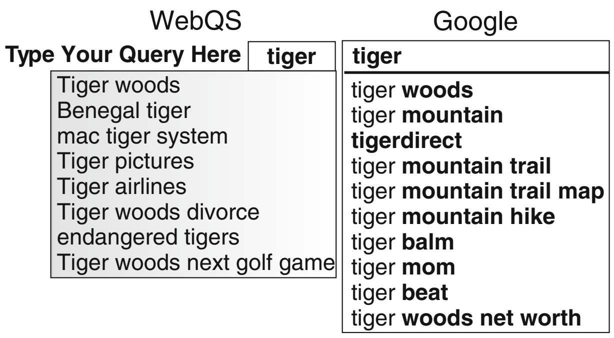
where

- $freq(SQ)$ is the *frequency of occurrence* of SQ in the AOL query logs.
- $WCF(Q, SQ)$, which is equal to $WCF(SQ, Q)$, is as defined in (1).
- $Mod(Q, SQ)$ is the *number of times* Q is *modified* to SQ in the same session in the AOL query logs within 10 min.
- Equation (2) is the *Stanford Certainty Factor* [20] on $freq$, WCF, and Mod. The Stanford Certainty Factor of an item I is a measure that integrates different scores or weights to yield an estimation of the *strength* of I. (The *Min* function is used when *all* the elements in the formula are required.)
- As $freq(SQ)$, $WCF(Q, SQ)$, and $Mod(Q, SQ)$ are in different numerical scales, prior to computing the *SuggRank* of SQ with respect to Q, they are *normalized* using a logarithmic scale to be in the same range.

WebQS selects the top 10 suggestions, which follows the 10 results per page approach employed by major web search engines.

Example 3 Figure 3 shows the queries suggested by WebQS (Google, respectively) for the keyword “tiger”, which is supposed to be entered by a web search engine user. Suggestions made by Google for “tiger” are mostly focused on “tiger mountain”, a mountain in the U.S. state of Washington, and “Tiger Woods”, a golf player, in

Figure 3 Queries suggested by WebQS and Google for the keyword query "tiger" posted on August 1, 2012, respectively



WebQS

Type Your Query Here tiger

Tiger woods
Benegal tiger
mac tiger system
Tiger pictures
Tiger airlines
Tiger woods divorce
endangered tigers
Tiger woods next golf game

Google

tiger

tiger **woods**
tiger **mountain**
tigerdirect
tiger **mountain trail**
tiger **mountain trail map**
tiger **mountain hike**
tiger **balm**
tiger **mom**
tiger **beat**
tiger **woods net worth**

addition to other suggested queries "tiger balm", "tiger mom", and "tiger beat", which are not intuitively clear what they refer to. The queries suggested by Google are likely based on the strategy adopted by Google which considers the *number of times* keywords in a query are submitted to the search engine as a feature in query suggestion and ranking, and "tiger mountain" and "Tiger Woods" were widely-used queries at that time. Compared with the queries suggested by WebQS, WebQS offers a wide variety of choices, which include animals or airlines named "tiger", besides "tiger mountain" and "Tiger Woods", as shown in Figure 3. The diversity of choices provided by WebQS enhances the chance that one of the suggested queries meets the information needs to be specified by the user.

4 Experimental results

We have assessed the overall performance of WebQS. In this section, we first detail the statistical approach that determines the (i) ideal *number of appraisers* and *queries* used for evaluating the query suggestion (QS) module of WebQS (in Section 4.1) and (ii) usefulness of QS (in Section 4.2). The performance analysis addresses the effectiveness of our query suggestion and ranking approaches (in Section 4.3.2). We have also measured the *processing time* of WebQS in suggesting queries (in Section 4.3.3) based on appraisers' evaluations.

4.1 Number of appraisers and test queries used for the controlled experiments

Prior to conducting the performance evaluation of WebQS and comparing its performance with other web QS modules, we first determine the *ideal* number of *appraisers* (in Section 4.1.1) and *test queries* (in Section 4.1.2) to be used in our empirical study so that the evaluation is sound, reliable, and objective.

4.1.1 The number of appraisers

In statistics, two types of errors, Type I and Type II, are defined [15]. Type I errors, also known as α errors or *false positives*, are the *mistakes* of *rejecting* a null hypothesis when it is true, whereas Type II errors, also known as β errors or *false negatives*, are the *mistakes* of *accepting* a null hypothesis when it is in fact false. We apply the following formula in [15] to determine the ideal number of *appraisers*, n, which is dictated by the probabilities of occurrence of Types I and II errors, in evaluating the query suggestion approach of WebQS:

$$n = \frac{(Z_{\frac{\alpha}{2}} + Z_{\beta})^2 \times 2\sigma^2}{\triangle^2} + \frac{{Z_{\frac{\alpha}{2}}}^2}{2} \tag{3}$$

where

1. $\triangle$ is the *minimal expected difference* in comparing WebQS with others, which is set to 1.0 in our study;
2. σ^2 is the *variance* of data, i.e., suggested keywords in our case, which is 3.85 in our study;
3. α denotes the *probability* of making a *Type I error*, which is 0.05 in our study;
4. β denotes the *probability* of making a *Type II error*, which is 0.20 in our study. Based on the value of β, we can determine the probability of a false null hypothesis to be correctly rejected, i.e., $1 - \beta$;
5. Z is the value assigned to the standard *normal distribution* of data, which are suggested keywords in our case. According to the standard normal distribution, when $\alpha = 0.05$, $Z_{\frac{\alpha}{2}} = 1.96$, whereas when $\beta = 0.20$, $Z_{\beta} = 0.84$.

To determine the values of $\triangle$ and σ^2 for evaluating the query suggestion approach of WebQS, we conducted an experiment using a randomly sampled 100 test queries from the AOL logs. We chose only 100 queries, since $\triangle$ and σ^2, which are computed on a *simple random sample*, do not change using a larger sample set of queries.

To compare WebQS with the QS module M of any web search engine using the 100 test queries, we set $\triangle$ to be 1. $\triangle = 1$ implies that we expect WebQS to be at least as good as M. In computing the variance,[5] i.e., σ^2, we calculated the *mean* among the number of keywords suggested by WebQS that are relevant to a query Q being constructed. Hereafter, we averaged the sum of the square difference between the mean and the actual number of relevant suggested keywords created for each one of the 100 test queries. We obtained 3.85, which is the value of σ^2.

The values of α and β are set to be 0.05 and 0.20, respectively, which imply that we have 95 % *confidence* on the correctness of our analysis and that the *power* (i.e., probability of avoiding false negatives/positives) of our statistical study is 80 %. $\alpha = 0.05$ is widely used, whereas 0.80 is a conventional value for $1 - \beta$, and a test with $\beta = 0.20$ is considered to be statistically powerful.

Based on the values assigned to the variables in (3), the *ideal* number of *appraisers* required in our study is 62, which is computed as follows.

$$n(Query\ Suggestion) = \frac{(1.96 + 0.84)^2 \times 2 \times 3.85}{1^2} + \frac{1.96^2}{2} \cong 62$$

Note that the values of $\triangle$, σ^2, α, and β directly influence the size of n. Moreover, the results collected from the n appraisers in the study are expected to be comparable with the results that are obtained by the actual population [15], i.e., web search engine users.

[5] *Variance*, which is the square of standard deviation, measures the average dispersion of the scores in a distribution.

4.1.2 The number of test queries

To determine the ideal number of *test queries* to be used in the controlled experiments, we rely on two different variables: (i) the *average attention span* of an adult and (ii) the *average number of search queries* that a person often creates in one session when using a web search engine. As verified in [25], the average attention span of an adult is between 20 to 30 min, and according to Jansen et al. [14], the average number of queries created in one session by a user on a web search engine is 2.8 $\sim$ 3. Based on these numbers, each appraiser was asked to evaluate WebQS using *three* queries. We consider *three* each to be an ideal number of queries, since evaluating *three* queries takes approximately 25 to 30 min, which falls in the time span of an adult. We randomly selected *186* (= 62 $\times$ 3) queries from the AOL query logs for the QS assessment.

4.2 Evaluation measures of WebQS

We have developed various applications on Facebook so that the 62 Facebook appraisers can assist us in evaluating WebQS using the 186 test queries. We chose appraisers from Facebook, since it is a social network with users diverse in ages, genders, nationalities, and cultures, who can provide objective and unbiased evaluations.

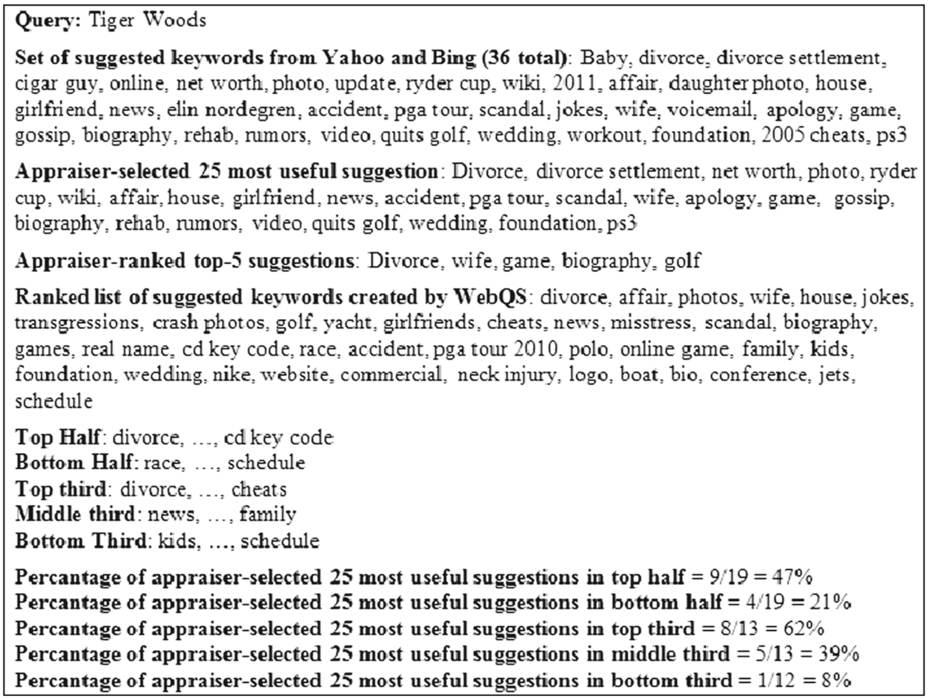
To evaluate suggested queries and their corresponding rankings for a test query Q offered by a number of QS modules, including WebQS, we follow the evaluation strategy introduced by Efthimiadis [8] who claims that users are the *best judge* in analyzing the performance of an interactive query suggestion system. Our evaluation compares the ranked queries provided by independent appraisers for Q, which are treated as the *gold standard*, with the ones offered by a QS module. The gold standard is employed to assess the usefulness of the suggestions and rankings provided by WebQS, Yahoo!, Bing, and a Baseline measure, respectively.[6] To accomplish this task, each appraiser, who was given a set of keyword queries suggested by Yahoo! and Bing on Q, was asked to select the 25 most useful ones and rank the top five (see Step 1 below), which yield the gold standard in terms of *useful* keywords and their *relative ranking positions* for Q. Detailed evaluation steps are given below.

1. Obtain the set of keyword queries suggested by Yahoo! and Bing for each one of the 186 test queries Q, respectively. Each appraiser who examines Q and its (combined) set of recommended keyword queries identifies the 25 most useful suggestions and determines the top-5 ranking. The gold standard (on the top-5 ranked, suggested queries) of Q is determined by the *highest occurrence* of the choices at each ranking position, i.e., from the 1st to the 5th, made by the 62 appraisers. If there is a *tie* at a particular top-5 ranking position, include each of them at the ranked position in the gold standard. If a suggested query SQ for Q has already been picked at a *higher* ranking position, then SQ *cannot* be chosen at a *lower* ranking position among the top-5 choices.
2. Process Q through WebQS (Yahoo!, Bing, and the Baseline measure, respectively) and obtain for Q the ranked list of suggested keyword queries.

[6]Google is not considered, since its query suggestion algorithm is not available to the public and thus cannot be implemented.

3. Divide the ranked list of suggested keyword queries created in Step 2 into two separated, ranked listings: 2 halves (*top* and *bottom half*) and 3 parts (*top third*, *middle third*, and *bottom third*).
4. Compare the appraiser's choices of suggested keyword queries and their rankings created in Step 1 for Q with each of the ranked lists L generated in Step 3 for Q. For each L, compute the *percentage* of keyword queries in L that are also chosen by the appraiser in Step 1, which yields a distribution of the appraiser's chosen keywords over L.
5. Based on the results generated in Steps 1 and 2, compute the respective *Normalized Discounted Cumulative Gain* (nDCG) values on the test queries for WebQS, Yahoo!, Bing, and the Baseline measure, respectively. Hereafter, use the *Wilcoxon signed-rank test*, which is a non-parametric test based on the differences between pairwise samples [6], to determine the *statistically significance* of the nDCG values achieved by WebQS with respect to the nDCG values achieved by Yahoo!, Bing, and the Baseline measure, respectively on the test queries.

The percentages of keyword queries included in each ranked list generated in Step 4 are averaged, which yields an estimation on the distribution of *useful keyword queries* suggested by each of the four QS modules. Figure 4 shows an example of the evaluation of keyword queries suggested and ranked by WebQS based on an appraiser's preference on the suggestions recommended by Yahoo! and Bing for


Query: Tiger Woods

Set of suggested keywords from Yahoo and Bing (36 total): Baby, divorce, divorce settlement, cigar guy, online, net worth, photo, update, ryder cup, wiki, 2011, affair, daughter photo, house, girlfriend, news, elin nordegren, accident, pga tour, scandal, jokes, wife, voicemail, apology, game, gossip, biography, rehab, rumors, video, quits golf, wedding, workout, foundation, 2005 cheats, ps3

Appraiser-selected 25 most useful suggestion: Divorce, divorce settlement, net worth, photo, ryder cup, wiki, affair, house, girlfriend, news, accident, pga tour, scandal, wife, apology, game, gossip, biography, rehab, rumors, video, quits golf, wedding, foundation, ps3

Appraiser-ranked top-5 suggestions: Divorce, wife, game, biography, golf

Ranked list of suggested keywords created by WebQS: divorce, affair, photos, wife, house, jokes, transgressions, crash photos, golf, yacht, girlfriends, cheats, news, misstress, scandal, biography, games, real name, cd key code, race, accident, pga tour 2010, polo, online game, family, kids, foundation, wedding, nike, website, commercial, neck injury, logo, boat, bio, conference, jets, schedule

Top Half: divorce, …, cd key code
Bottom Half: race, …, schedule
Top third: divorce, …, cheats
Middle third: news, …, family
Bottom Third: kids, …, schedule

Percantage of appraiser-selected 25 most useful suggestions in top half = 9/19 = 47%
Percantage of appraiser-selected 25 most useful suggestions in bottom half = 4/19 = 21%
Percantage of appraiser-selected 25 most useful suggestions in top third = 8/13 = 62%
Percantage of appraiser-selected 25 most useful suggestions in middle third = 5/13 = 39%
Percantage of appraiser-selected 25 most useful suggestions in bottom third = 1/12 = 8%

Figure 4 An example of evaluating keyword queries suggested by WebQS

the query "Tiger Woods". The *higher* the percentage on the top-half (top third, respectively) is, the *more effective* the query suggestion and ranking of WebQS is with respective to the query.

Using the results generated in Steps 1 and 2, we compute in Step 5 the nDCG values, denoted nDCG$_5$, on the suggestions and rankings provided by the each of the four QS modules. nDCG, i.e., nDCG$_5$ as defined in (4), penalizes a top-5 suggestion, i.e., a suggested keyword query in the gold standard, that is ranked by a QS module *lower* in its list of top-5 suggested queries. The penalization is based on a relevance reduction, which is logarithmically proportional to the position of each suggested keyword query in the list (as shown in (5)). The *higher* the nDCG score is, the *better* the query suggestion and ranking strategy adopted by the corresponding QS module is, since it positions useful suggestions *higher* in the generated ranking.

$$nDCG_5 = \frac{1}{N}\sum_{i=1}^{N}\frac{DCG_{5,i}}{IDCG_{5,i}} \tag{4}$$

where N is the total number of test queries, which is 186, $IDCG_{5,i}$ (the *Ideal* Discounted Cumulative Gain) is the best possible $DCG_{5,i}$ value for the ranked suggestions for the ith test query,[7] and

$$DCG_{5,i} = rel_{i_1} + \sum_{j=2}^{5}\frac{rel_{i_j}}{log_2 i_j} \tag{5}$$

where rel_{i_k} is the *graded relevance level* of the kth ranked suggested query for the i^{th} test query. The graded relevance level on a six-point scale, i.e., $0 \leq rel_{i_k} \leq 5$, $1 \leq k \leq 5$, is widely-used, which we have adopted for our query suggestion and ranking performance evaluation such that $rel_{i_k} = 0$ if the corresponding kth ranked suggestion is not among the top-5 suggestions specified in the gold standard of the ith test query, and $rel_{i_k} = 5$ if the corresponding suggested query is the *first* suggested query, and so on.

4.3 Performance evaluations

We quantify the *effectiveness* of the QS module of WebQS in recommending useful query keywords, which are determined by the Facebook appraisers involved in our controlled experiment.

4.3.1 Other QS modules and the ranking strategy

We have compared the performance of WebQS with Yahoo!, Bing, and a Baseline measure using the 186 test queries, respectively. The Baseline measure recommends suggested queries solely based on the *frequencies of queries* in query logs, i.e., the number of times an original user query is modified to the suggested ones, whereas Yahoo! and Bing apply their individual algorithms to create a ranked list of suggested queries, respectively. We detail the design of their respective QS module below.

[7] $IDCG_{5,i}$ is computed using an *ideal* ranking on $DCG_{5,i}$ such that the suggestions are arranged in descending order on their graded relevant judgment scores, i.e., rel_{i_j}, $1 \leq j \leq 5$, from 5 to 1 in $DCG_{5,i}$.

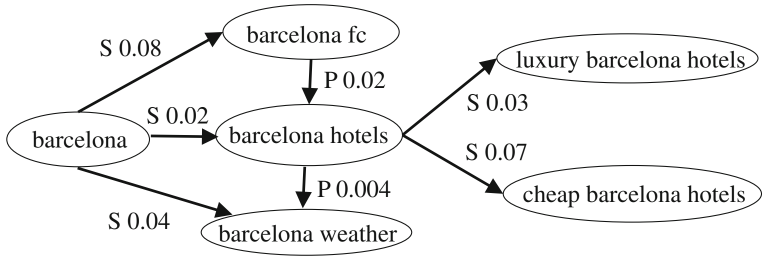


Figure 5 An excerpt of the query flow graph for the query "barcelona", created by using the Yahoo! UK query log

Yahoo!: The Yahoo! search engine recommends query keywords by using *query flow graphs* [3]. A query flow graph is an aggregated representation of the latent querying behavior contained in a query log. Intuitively, in a query flow graph, a directed edge from query q_i to query q_j denotes that the two queries are part of the same search session. Any nodes on a path in a query flow graph may be seen as potential suggested queries whose likelihood is given by the *strength* of the edges along the path. A directed edge (q_i, q_j) is labeled with (i) the *probability* that a user creates q_j after q_i in the same search session (i.e., q_j is a possible suggestion to q_i), and (ii) the type of the transition, known as the *Query Reformulation Type* (QRT). QRT indicates the relationship between two queries, q_i and q_j, linked by a directed edge in a query flow graph, which include *Equivalent Rephrasing* (i.e., q_j has the same meaning as q_i but is rephrased differently), *Parallel Move* (i.e., q_i and q_j are two subtopics of the user query), *Generalization* (i.e., q_j is a generalization of q_i), *Specialization* (i.e., q_j is a special case of q_i), etc. In a query flow graph created by the Yahoo! search engine, edges are directed and are annotated with two labels, the QRT and a *weight*, as given by a pre-trained model.[8] The weight of queries q and q' is defined as

$$w(q, q') = \frac{r(q, q')}{\sum_{k:(q,k)\in E} r(q, k)} \tag{6}$$

where $r(q, q')$ denotes the number of times q has been directly modified to q' as shown in a query log, and the weight $w(q, q')$ represents the *probability* of q and q' being in the same querying session. A small excerpt of a query flow graph is shown in Figure 5, where S and P denote the *Specialization* and *Parallel Move* query reformulation types, respectively.

Yahoo! retrieves the top-n nodes (queries) based on their edge weights in a query flow graph such that the chosen nodes, which are suggested queries, have the *highest edge weights* among all the nodes that are either directly connected to a user query node or any indirect nodes that are connected to a node already retrieved as a suggestion, where n is set to be 10 by Yahoo!. As shown in Figure 5, given the user search query "barcelona", the query suggestions retrieved in order

[8]A group of Yahoo! editors manually labeled the set of query pairs (q, q') in each search session using one of the reformulation types to create the training dataset.

are “Barcelona fc” (0.08), “Barcelona weather” (0.04), “Barcelona hotels” (0.02), “cheap Barcelona hotels” (0.07), and “luxury Barcelona hotels” (0.03).

Bing: MSN search (now known as Bing) constructs a bipartite graph using a query log[9] and computes *hitting time* (i.e., click frequency) [11] to determine query keywords to be recommended for a user query. A bipartite graph $G = (V_1 \cup V_2, E)$ consists of a query set V_1, which is the set of queries extracted from the query log, and a URL set V_2, which is the set of URLs specified in the log. An edge in E from a query i to an URL k denotes that k was clicked by the user who submitted i to Bing, and the edge is *weighted* by the *click frequency*, denoted $w(i, k)$, which indicates the number of times k is clicked when i is the search query. During the query suggestion phase, a modified subgraph SG is constructed from G using the depth-first search approach on a user query Q with queries in V_1. SG is a modified subgraph of G, since each URL node k, which is in the original G, is removed from SG, but the connected edges of k are retained. The search of suggested queries for Q stops when the number of nodes representing suggested queries on SG is larger than a predefined number of n queries, which is set by Bing to be 10. Hereafter, Bing defines the *transition probability*, $P_{i,j}$, between any two queries, i and j, linked by an edge in SG, which computes the degree of similarity between i and j as

$$P_{i,j} = \sum_{k \in V_2} \frac{w(i,k)}{d_i} \times \frac{w(j,k)}{d_j} \tag{7}$$

where k is a URL in G, $d_i = \sum_{k \in V_2} w(i,k)$, and $d_j = \sum_{k \in V_2} w(j,k)$. For all the queries exhibited in SG, excluding the one being searched, Bing retrieves the queries with the top-n *largest* transition probabilities with respect to query i as suggestions.

Baseline: A query suggestion approach is often compared against a Baseline measure, which ranks query suggestions strictly based on their *frequencies of occurrence* in a query log. The list of queries suggested by a Baseline measure is created by retrieving all queries in the query log that include the user query Q as a *substring*. The more *frequent* a query suggestion S, which includes Q as a substring in the query log is, the *higher* S is ranked.

4.3.2 The effectiveness measures

We have collected the evaluations compiled by using the ranked keyword queries suggested by WebQS (Yahoo!, Bing, and the Baseline measure, respectively) that were compared against the gold standard, i.e., the top-5 ranked keyword queries, established by the 62 Facebook appraisers for each one of the 186 test queries. Based on the evaluations, we computed (i) the averaged percentage on the occurrence of suggested keyword queries for each test query on each of the five parts, i.e., top-half, bottom-half, top-third, middle-third, and bottom-third, and (ii) the nDCG values for the suggestions made by WebQS (Yahoo!, Bing, and the Baseline measure, respectively) on all the 186 test queries. The percentages achieved by WebQS were

[9]In performing the comparisons with WebQS, we used the same AOL query logs on each of the three query suggestion systems, i.e., Bing, Yahoo!, and the Baseline measure.

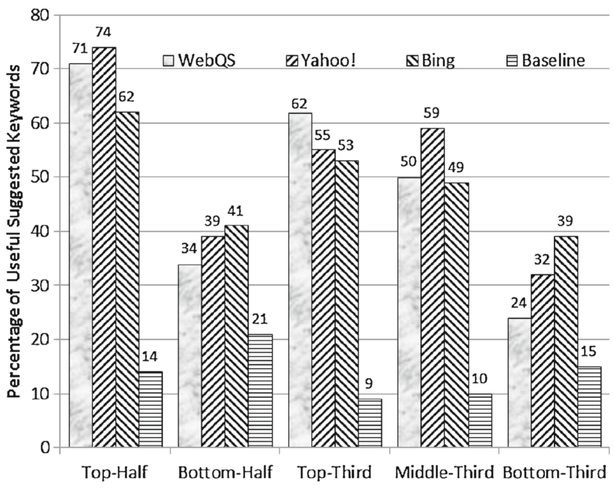


Figure 6 Averaged percentages of useful ranked queries suggested by WebQS, Yahoo!, Bing, and the Baseline measure, respectively computed using Facebook appraisers' evaluations

compared against the ones achieved by Yahoo!, Bing, and the Baseline measure, respectively, whereas the nDCG values were used in the Wilcoxon signed-rank test.

Figure 6 shows that WebQS achieves a relatively high percentage over the *top-half* and *top-third* and a relatively low percentage on the *bottom-half* and *bottom-third*, which imply that keyword queries suggested and ranked by WebQS are useful, since they are consistent with the choices, in terms of suggestions and rankings, made by Facebook appraisers. Although Yahoo! achieves a slightly higher percentage than WebQS over the *top-half*, WebQS obtains a much higher percentage on the *top-third*, which indicates that useful keywords are often ranked higher by WebQS than by Yahoo! (Bing and the Baseline measure, respectively). Figure 7, on the other hand, shows the nDCG score achieved by each of the four QS models: Yahoo!, Bing, the Baseline measure, and WebQS.

According to the Wilcoxon test (with $p < 0.001$), WebQS outperforms the Baseline measure for query suggestions, since the improvement in the nDCG values achieved by the former is *statistically significant* than the ones achieved by the latter. Furthermore, neither Yahoo! nor Bing outperforms WebQS, since the improvement

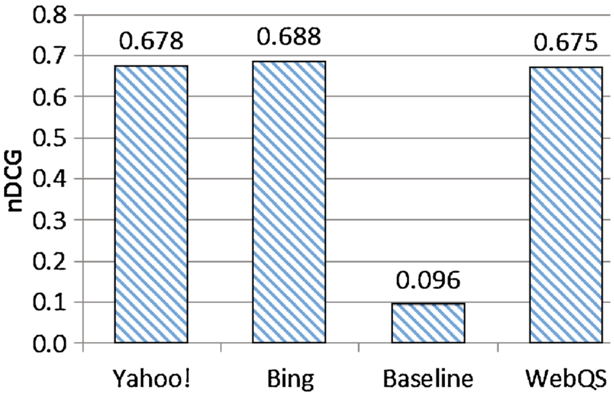


Figure 7 The nDCG scores for Yahoo!, Bing, the *Baseline measure*, and WebQS, respectively based on the rankings of suggested queries compiled by Facebook appraisers

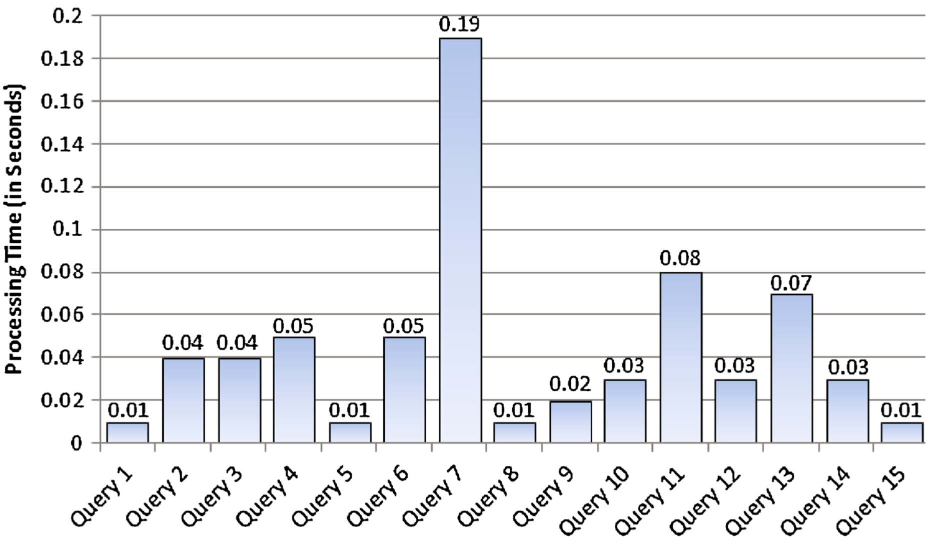


Figure 8 Processing time on query suggestions for 15 (out of 186) queries imposed on WebQS

in terms of the nDCG scores achieved by Yahoo! or Bing over WebQS's is not statistically significant (with $p < 0.01$). By analyzing the statistically significance of the nDCG scores achieved by WebQS and its counterparts, we can claim that WebQS is *better* than the Baseline measure approach and is as *good* as Yahoo! and Bing as a query suggestion tool.

4.3.3 Processing time of WebQS

We have also measured the required *processing time* of the query suggestion approach of WebQS, Google, Yahoo!, and Bing, respectively using the 186 test queries. The time required to suggest queries by WebQS for a user's input is on an average of 0.07 seconds, which indicates that WebQS generates query suggestions *instantly* while the user is formulating his/her query, and is comparable to Google, Yahoo!, and Bing. Figure 8 shows the processing time of WebQS for making suggestions on 15 selected (out of the 186 test) queries.

5 Conclusions

Web search engines have become part of our daily lives these days which provide a valuable source of information for people from all walks of life without boundary of age nor educational background. Current web search engines, such as Google, Yahoo!, and Bing, offer users a mean to locate desired information available on the Web. Each of these engines is equipped with a query suggestion module, which suggests keywords to be used while its users is entering search queries. The algorithms of these query suggestion modules, however, are either not publicly available or rely on the currently-searched words to suggest query keywords. In solving these problems, we have developed a web search query suggestion system, denoted

WebQS, which assists its users in *formulating* their queries using a simple, yet effective trie-based query suggestion module to enhance the precision of web search. The development of WebQS is a contribution to the web search community, since based on the detailed design of WebQS as presented in this paper, search engine developers can adopt and implement WebQS on a platform to conduct the type of searches, such as desktop search or vertical search, as needed.

The design of the query suggestion approach of WebQS simply considers the *frequency* of query *co-occurrence*, *word similarity*, and *modified* queries, which is *unique* and *elegant*. In addition, WebQS detects spelling errors among keywords entered by users to provide further assistance to its users in formulating search queries.

Results of the conducted empirical study have showed that WebQS is effective, since it performs as well as Yahoo! and Bing in query suggestion and ranking and is comparable, in terms of efficiency, to the query suggestion approaches of the most popular web search engines these days.

References

1. Baraglia, R., Castillo, C., Donato, D., Nardini, F., Perego, R., Silvestri, F.: The effects of time on query flow graph-based models for query suggestion. In: Proceedings of RIAO'10: Adaptivity, Personalization and Fusion of Heterogeneous Information, pp. 182–189 (2010)
2. Bhatia, S., Majumdar, D., Mitra, P.: Query suggestions in the absence of query logs. In: Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 795–804 (2011)
3. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Vigna, S.: Query suggestions using query flow graphs. In: Proceedings of the ACM Workshop on Web Search Click Data (WSCD), pp. 56–63 (2009)
4. Cao, G., Nie, J., Bai, J.: Integrating word relationships into language models. In: Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 298–305 (2005)
5. Cao, H., Jiang, D., Pei, J., He, Q., Liao, A., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 875–883 (2008)
6. Croft, B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice. Addison Wesley (2010)
7. Duan, H., Hsu, B.-J.: Online spelling correction for query completion. In: Proceedings of World Wide Web (WWW), pp. 117–126 (2011)
8. Efthimiadis, E.: Interactive query expansion: a user-based evaluation in a relevance feedback environment. J. Am. Soc. Inf. Sci. (JASIS) **51**, 989–1003 (2000)
9. Fonseca, B., Golgher, P., Possas, B., Ribeiro-Neto, B., Ziviani, N.: Concept-based interactive query expansion. In: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), pp. 696–703 (2005)
10. Gao, J., Li, X., Micol, D., Quirk, C., Sun, X.: A large scale ranker-based system for search query spelling correction. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling), pp. 358–366 (2010)
11. Graepel, T., Candela, J., Borchert, T., Herbrich, R.: Web-scale Bayesian click-through rate prediction for sponsored search advertising in microsoft's Bing search engine. In: Proceedings of the 27th International Conference on Machine Learning (ICML'10), pp. 13–20 (2010)
12. Hansan, M., Parikh, N., Singh, G., Sundaresan, N.: Query suggestion for e-commerce sites. In: Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM), pp. 765–774 (2011)
13. Huang, J., Efthimiadis, E.: Analyzing and evaluating query reformulation strategies in web search logs. In: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), pp. 77–86 (2009)
14. Jansen, B., Spink, A., Saracevic, T.: Real life, real users, and real needs: a study and analysis of user queries on the web. Inf. Process. Manag. (IPM) **36**(2), 207–227 (2000)

15. Jones, B., Kenward, M.: Design and Analysis of Cross-Over Trials, 2nd edn. Chapman and Hall (2003)
16. Kato, M., Sakai, T., Tanaka, K.: Structured query suggestion for specialization and parallel movement: effect on search behaviors. In: Proceedings of International Conference on World Wide Web (WWW), pp. 389–398 (2012)
17. Koberstein, J., Ng, Y.-K.: Using word clusters to detect similar web documents. In: Proceedings of the 1st International Conference on Knowledge Science, Engineering and Management (KSEM), pp. 215–228 (2006)
18. Liao, Z., Jiang, D., Chen, E., Pei, J., Cao, H., Li, H.: Mining concept sequences from large-scale search logs for context-aware query suggestion. ACM Transactions on Intelligent Systems and Technology (ACMTIST) **3**(1), 17:1–17:40 (2011)
19. Liu, S., Liu, F., Yu, C., Meng, W.: An effective approach to document retrieval via utilizing WordNet and recognizing phrases. In: Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 266–272 (2004)
20. Luger, G.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley (2008)
21. Mei, Q., Zhou, D., Church, K.: Query suggestion using hitting time. In: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), pp. 469–478 (2008)
22. Oliver, J.: Decision graphs—an extension of decision trees. Technical Report 92/173, Monash University (1992)
23. Pera, M., Ng, Y.-K.: SimPaD: a word-similarity sentence-based plagiarisum detection tool on web documents. Web Intelligence and Agent Systems: An International Journal (WIAS) **9**(1), 27–41 (2011)
24. Qumsiyeh, R., Ng, Y.-K.: ReadAid: a robust and fully-automated readability assessment tool. In: Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 539–546 (2011)
25. Rozakis, L.: Test Taking Strategies and Study Skills for the Utterly Confused. McGraw Hill (2002)
26. Ruch, P., Tbahriti, I., Gobeill, J., Aronson, A.: Argumentative feedback: a linguistically-motivated term expansion for information retrieval. In: Proceedings of International Conference on Computational Linguistics (COLING), pp. 675–682 (2006)
27. Song, Y., He, L.: Optimal rare query suggestion with implicit user feedback. In: Proceedings of International Conference on World Wide Web (WWW), pp. 901–910 (2010)
28. Strube, M., Ponzetto, S.: WikiRelate! Computing semantic relatedness using wikipedia. In: Proceedings of the 21st AAAI Conference on Artificial Intelligence, pp. 1419–1424 (2006)
29. Vectomova, O., Wang, Y.: A study of the effect of term proximity on query expansion. Inf. Sci. **32**(4), 324–333 (2006)