2013 IEEE 25th International Conference on Tools with Artificial Intelligence

A Probabilistic Query Suggestion Approach Without Using Query Logs

Meher T. Shaikh
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
Email: mehr_mujahed@yahoo.com

Maria S. Pera
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
Email:soledadpera@gmail.com

Yiu-Kai Ng
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
Email:ng@compsci.byu.edu

***Abstract*—Commercial web search engines include a query suggestion module so that given a user's keyword query, alternative suggestions are offered and served as a guide to assist the user in formulating queries which capture his/her intended information need in a quick and simple manner. Majority of these modules, however, perform an in-depth analysis of large query logs and thus (i) their suggestions are mostly based on queries frequently posted by users and (ii) their design methodologies cannot be applied to make suggestions on customized search applications for enterprises for which their respective query logs are not large enough or non-existent. To address these design issues, we have developed PQS, a probabilistic query suggestion module. Unlike its counterparts, PQS is not constrained by the existence of query logs, since it solely relies on the availability of user-generated content freely accessible online, such as the Wikipedia.org document collection, and applies simple, yet effective, probabilistic- and information retrieval-based models, i.e., the Multinomial, Bigram Language, and Vector Space Models, to provide useful and diverse query suggestions. Empirical studies conducted using a set of test queries and the feedbacks provided by Mechanical Turk appraisers have verified that PQS makes more useful suggestions than Yahoo! and is almost as good as Google and Bing based on the relatively small difference in performance measures achieved by Google and Bing over PQS.**

***Keywords*-Query suggestion; classification; probabilities**

I. Introduction

To conduct a web search, users rely heavily on the keyword search option offered by existing web search engines, such as Google, Bing, and Yahoo!, to create queries that can be very broad to very specific in nature. As some of these queries are short and ambiguous, it is difficult for search engines to discern the actual information needs of their users. Moreover, a significant number of users lack knowledge on how to formulate proper queries for difficult topics [1]. In solving these problems and aiding users in their quest for online resources, widely-used web search engines have incorporated a query suggestion (QS) module, which recommends useful queries in response to users' inputs.

Most of the existing QS approaches rely on the "wisdom of crowds" and perform an in-depth analysis of query logs based on either graph-based models [2] or probabilistic models [3], which consider co-occurred queries, term associations patterns, and/or web results clicked/ignored by users in query log sessions [3]. These QS systems, however, (i) depend on the availability of *very large* query logs, which are not always publicly and freely accessible due to privacy and legal constraints, (ii) may have limited coverage, since they cannot infer from past observations queries that have not (co-)occurred in the logs [4], and (iii) cannot be applied to make suggestions on customized search tools developed for applications that are used by a small number of users (with few hundreds or thousands of users) [5]. To address these major design issues, we have developed PQS, a probabilistic QS approach.

PQS makes suggestions without using query logs. Instead, it relies on Wikipedia documents and well-established probabilistic models to (i) identify the *categories*, i.e., topics of information, addressed on a user-initiated keyword query Q and (ii) generate candidate query suggestions that correlate with Q. For each candidate suggestion CS, PQS computes its *ranking score* by employing a mathematically-sound aggregation strategy [6] on a number of measures that capture (i) the *likelihood* of CS, which matches the search intent of an individual user to the pre-defined information category c to which CS belongs, (ii) the *probability* of generating CS based on the bigram co-occurrence of terms in c, and (iii) the *degree of similarity* between CS and the word-probability distribution of c. Hereafter, the top-ranked queries ordered according to their ranking scores are provided to the user as suggestions for Q.

PQS requires neither pre-defined data models, such as ontologies, nor user-feedback to make query suggestions. Moreover, PQS, which is not restricted in coverage, enhances users' web search experience by *explicitly considering* the potential *multiple interpretations* of keywords specified in a user-initiated query. Unlike its document-centric counterpart in [5], which relies on designated document corpus from specific applications to suggest queries, PQS simply applies well-known and elegant probabilistic and information retrieval models, i.e., the Multinomial, Bigram Language, and Vector Space Models, which are trained using the Wikipedia document collection that covers a variety of topics, to make suggestions for diverse applications using the same trained models.

The remaining of this paper is organized as follows. In

Table I
CATEGORIES USED BY PQS FOR QUERY CLASSIFICATION

Animals	Architecture	Art	Aviation	Beliefs
Books	Business	Cars	Cities	Crime
Data Viz	Design	Drugs	Economics	Education
Entertainment	Food	Funny	Gender	Health
Histories	Internet	Maps	Media	Movies
Music	Nature	News	Politics	Science
Space	Sports	Style	Technology	TV
Video	Warfare	World		

Section II, we introduce PQS. In Section III, we present the results of the empirical study conducted to evaluate PQS and compare its performance against the performance of widely-used, popular search engines. In Section IV, we give a concluding remark.

II. OUR PROBABILISTIC QUERY SUGGESTION MODULE

In this section, we detail the design methodology of PQS in making suggestions for a user query.

A. *Diverse Query Suggestions Based on Categories*

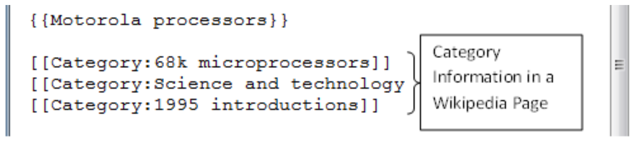
A keyword query is generally short and ambiguous [7] and belongs to one or more topics or domains. Uniquely identifying the information need of an ambiguous query, which often consists of only one or two keywords, is an undecidable problem. To address this issue, PQS accounts for diverse search intents and offers suggestions belonged to different domains.

PQS considers the categories extracted from Digg.com[1] and assigns a collection of Wikipedia documents to their corresponding categories. We consider the Digg's categories instead of the category information specified in Wikipedia documents, since there are 14,615 distinct (sub-)categories specified in Wikipedia documents with majority of them being very specific. We prefer more general categories, since web queries are rarely specific, which is confirmed by the study presented in [8] that demonstrates a strong correlation between decreasing query length and increasing generality of a user's information need. PQS considers 37 out of the 48 categories defined by Digg, while the remaining Digg's categories, including "long reads", "lust", and "booze", are excluded, since they are uncommon and seldom matched by the Wikipedia category information. Table I shows the list of all the 38 categories considered by PQS, which include "entertainment", a popular category, in addition to the 37 categories defined in Digg.

To capture the content of each of the 38 categories, we randomly selected 10,000 Wikipedia documents for each category from the Wikipedia document dump[2] based on the category information specified in each document. Figure 1 shows the Wikipedia document, 'Freescale Dragonball', which belongs to the Digg's categories "science" and "technology". To assign a Wikipedia document D to a category,

[1]Digg.com is a social world news website.

[2]http://dumps.wikimedia.org/enwiki/20130204/



```
{{Motorola processors}}

[[Category:68k microprocessors]]
[[Category:Science and technology
[[Category:1995 introductions]]
```

Figure 1. Category Information of a Wikipedia Document

we match its category information against the 38 categories employed by PQS (identified by unique names) based on either the exact match or similarity match[3] of the keywords specified in the category information in D and the category names. For example, the Wikipedia document as shown in Figure 1 is assigned to two different categories, "science" and "technology".

B. *The Probabilistic Models*

In making query suggestions, PQS employs two different probabilistic models, i.e., the *Multinomial* and *Bigram Language* Models, which are presented below.

1) The Multinomial Model: PQS makes suggestions partially based on the probability statistics of observing a keyword in a category. One of the popular models to determine the probability of occurrence of a keyword in a category is the multinomial model [10]. To capture the keyword probability distribution in a given category c (among the 38 categories considered by PQS), we train a multinomial model using 3,000 out of the 10,000 Wikipedia documents assigned to each category as discussed in Section II-A. Based on the *frequency of occurrence* of the non-stop, stemmed[4] keywords in c, we define the probability of observing a keyword k in c using the Laplacian Smoothed Estimation [10] as

$$P(k \mid c) = \frac{tf_{k,c} + 1}{|c| + |V|} \tag{1}$$

where $|c|$ is the number of non-stop, stemmed keywords in the 3,000 training documents of c, $|V|$ is the number of distinct non-stop, stemmed keywords in the 114,000 (= 3,000 $\times$ 38) Wikipedia documents used for training, and $tf_{k,c}$ is the frequency of occurrence of k in documents of c.

2) The Bigram Language Model: To identify candidate suggestions for a user query Q and estimate the likelihood of occurrence of a created candidate suggestion (as described in Sections II-C and II-E, respectively), PQS relies on a bigram language model that provides probability statistics of consecutive term occurrences within each category. We define a *term* as a non-stopword keyword, which can be preceded by a sequence of *connection words*.[5] Connection words [5] are stopwords in a term that are *not* counted as

[3]The word-correlation factor matrix defined in [9] is used to determine words similar to a given word.

[4]Keywords in the Wikipedia documents are reduced to their grammatical roots using the commonly-used Porter Stemmer.

[5]A connection word is either an article, a preposition, or a conjunction.

words in the term but retained in the term to capture the precise meaning of a suggested query. For example, if a user enters the query keyword ‘symptoms’, a suggestion ‘symptoms of pneumonia’ makes more sense than the suggestion ‘symptoms pneumonia’, since in the latter case the relationship between the two keywords is missing.

During the process of training the bigram language model, we examine the consecutive term occurrences in the 10,000 Wikipedia documents of each pre-defined category c. Using these term occurrences, we determine $P(t_i|t_{i-1})$, the probability of term t_i following term t_{i-1} (in c), as

$$P(t_i|t_{i-1}) = \frac{f_{t_i,t_{i-1}} + 1}{|b_c| + |V_b|} \qquad (2)$$

where $f_{t_i,t_{i-1}}$ is the number of times t_{i-1} is followed by t_i in the sentences of documents in c, b_c is the total number of bigrams in c, and V_b is the distinct number of bigrams in c.

In computing $P(t_i|t_{i-1})$, we consider a special case, i.e., $P(\text{EOS}|t_{i-1})$, where EOS stands for “End Of Sentence”. $P(\text{EOS}|t_{i-1})$ captures the probability of term t_{i-1} being the *last* term in a sentence. PQS considers all the terms following t_{i-1}, including EOS[6].

We consider a bigram language model so that searching for related terms to a user query becomes *more efficient* than maintaining the probabilities of sequences of more than two terms which increases the database size [5] for the document collection and significantly impacts the search time for terms related to a given query.

C. Candidate Suggestions

When a user enters a keyword query Q, PQS first generates a set of n-gram suggestions such that each n-gram (n $>$ 1) extracted from across different pre-defined categories in response to Q is treated as a *candidate suggestion*.

To determine the candidate suggestions for Q, PQS examines the probabilities of occurrence of terms that follow the last term in Q (in diverse categories) using the bigram model presented in Section II-B2. Given Q with m (≥ 1) terms and a category c, PQS identifies the top-five[7] t_{m+1} terms that are the most frequent terms following t_m, where m is last term in Q and $m+1$ is the first suggested term for Q, based on their respective $P(t_{m+1}|t_m)$ values. For each one of the top-five terms t_{m+1} in c, PQS considers the next top-five $P(t_{m+2}|t_{m+1})$ and compares them with $P(\text{EOS}|t_{m+1})$ to determine whether the corresponding t_{m+2} are either retained or ignored.[8] If $P(\text{EOS}|t_{m+1})$ is the *largest* among all the probabilities associated with the top-5 t_{m+2}, i.e., $P(t_{m+2}|t_{m+1})$, then it is an evidence that t_{m+1} is more likely to be followed by an EOS than any other term. Consequently, t_{m+1} is treated as the last term in the candidate suggestion currently being considered, and other n-grams with terms following t_{m+1} are excluded as candidate suggestions. Otherwise, the next set of top-five terms following t_{m+2} is examined, and so on.

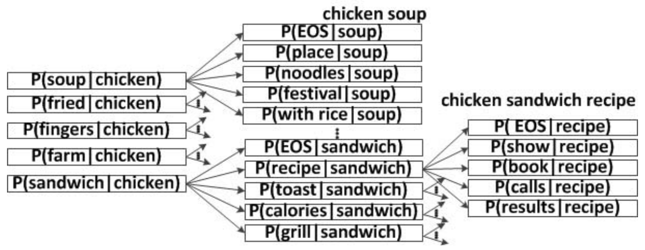


Figure 2. Candidate suggestions for the query “chicken” within the *food* category using the Bigram Language Model defined in Section II-B2

Example 1: Figure 2 shows sample candidate suggestions in the *food* category. Given the query keyword “chicken”, the top-five terms that follow “chicken” in the *food* category are “soup”, “fried”, “fingers”, “farm”, and “sandwich”. Among the probability values of the top-5 terms that follow “soup”, $P(\text{EOS}|\text{soup})$ turns out to be the largest, and hence other n-grams with prefix “chicken soup” are not further considered as candidate suggestions. However, $P(\text{EOS}|\text{sandwich})$ is not the largest among the probabilities of terms that follow “sandwich”, and hence other additional terms, such as “recipe”, “toast” and “calories”, are further considered as trigram suggestions, and so on. □

D. Category Likelihood Score of a Candidate Suggestion

PQS computes the likelihood values of different categories given a user query Q with m (≥ 1) terms, as shown in Equation 3, based on the assumption that the more frequently terms in Q appear in a category c, the more promising candidate suggestions originated from c are for Q. In accomplishing this task, PQS employs the multinomial model introduced in Section II-B1 along with the well-known Bayes’ rule [10] and compares the probability distribution of terms in different categories with the terms in Q.

$$P(c|Q) = \frac{\prod_{i=1}^{m} P(k_i|c)P(c)}{\sum_{c \in C} \prod_{i=1}^{m} P(k_i|C=c)P(C=c)} \qquad (3)$$

where $P(c)$ is the probability of observing c, which is the ratio of documents in c to the total number of documents used to train the multinomial model introduced in Section II-B1, C is the set of pre-defined 38 categories considered by PQS, and $P(k_i|c)$ is the probability that the i^{th} term in Q is observed in c as determined using the multinomial model.

PQS treats the likelihood value computed in Equation 3 as one of the measures to determine the significance of each

[6]Technically, EOS is not a term according to its definition. However, we consider the probability of $P(\text{EOS}|t_i)$ as an exceptional case in PQS.

[7]The design of PQS calls for at most 10 suggestions to be made for a user’s query Q. Given that less likely suggestions originated from less frequent terms will not make it to the ranked list of suggested queries for Q, PQS considers only the top-5 terms following Q (in each pre-defined category), which yields at least 190 (= 5 $\times$ 38) candidate suggestions.

[8]PQS does not examine $P(\text{EOS}|t_m)$, since each candidate suggestion must include at least one suggested term.

candidate suggestion CS and computes the *category likelihood* score of CS with respect to c, denoted $CL(CS, c)$.

$$CL(CS, c) = P(c|Q). \tag{4}$$

E. N-gram Probability Score of a Candidate Suggestion

As previously stated, candidate suggestions are n-grams generated from across different categories constructed based on the probabilities of term co-occurrences, i.e., bigram probabilities computed using the model introduced in Section II-B2, within the respective categories. Given a candidate suggestion CS, which is a sequence of terms, t_1, t_2, ..., t_n ($n > 1$) extracted from category c, such that t_i ($1 \leq i \leq m$) is one of the m (≥ 1) terms included in the initial query Q and t_i ($m+1 \leq i \leq n$) is one of the $n-m$ suggestion terms in CS, PQS computes the probability of generating such sequence using the following equation:

$$P(t_1, t_2, \ldots, t_n) = \frac{P(t_2|t_1) + P(t_3|t_2) + \ldots + P(t_n|t_{n-1})}{n-1} \tag{5}$$

where $P(t_i|t_{i-1})$, $2 \leq i \leq n$, is the probability of term t_i following term t_{i-1} in c. In Equation 5, the probabilities of a sequence of co-occurring terms are $averaged$, since an average is a simple mechanism to summarize the general significance of a set of unequal probabilities. The probability value computed in Equation 5 for CS is its n-gram probability score, denoted $NGS(CS, c)$, which is defined as

$$NGS(CS, c) = P(t_1, t_2, \ldots, t_n) \tag{6}$$

A high NGS value indicates that, in general, CS includes highly co-occurring bigrams, which means that CS is more likely treated as a useful suggestion for Q.

F. Degree of Cohesiveness of a Candidate suggestion

Given an n-gram candidate suggestion CS obtained from a category c, PQS analyzes the distribution of terms in CS across (the documents in) c. If all the terms in CS frequently occur in c, then they capture the content of a significant number of documents in c and are considered to be coherent with the information covered in c to a certain degree. To measure the *degree of cohesiveness* of CS with respect to c, PQS computes $Sim(CS, c)$, which reflects the *closeness* of the distributions of terms in CS and c using the well-known vector space model and the cosine similarity measure.

$$Sim(CS, c) = \frac{\sum_{i=1}^{V} c_i \times CS_i}{\sqrt{\sum_{i=1}^{V} c_i^2 \times \sum_{i=1}^{V} CS_i^2}} \tag{7}$$

where CS and c are represented as V-dimensional vectors of term probabilities, V is number of distinct terms defined for the multinomial model introduced in Section II-B1, c_i is the probability of observing term i in c determined by the multinomial model, and CS_i is the *weight* of term i in CS which is 1 if the term is in CS and is 0, otherwise.

If some of the terms in CS seldom occur in the documents of c, then CS is *not* a coherent suggestion as reflected by $Sim(CS, c)$.

Example 2: Consider two candidate suggestions from the *cars* category, "grand prix" and "grand slam". The term distribution of "grand", "prix", and "slam" across the *cars* category is 0.001337, 0.0012, and 0.0000045 respectively. Based on Equation 7, Sim("grand prix"$, cars)$ = 0.99 and Sim("grand slam"$, cars)$ = 0.71, which demonstrates that "slam" in the *cars* category occurs *less* frequently than "prix", and "grand slam" should be treated as a *less* useful suggestion than "grand prix" for the query "grand" in the *cars* category based on their degrees of cohesiveness. □

G. Ranking

Having computed the *category likelihood* score, i.e., $CL(CS, c)$, the *n-gram probability* score, i.e., $NGS(CS, c)$, and the *degree of cohesiveness*, i.e., $Sim(CS, c)$, for each candidate suggestion CS in category c, PQS ranks the candidate suggestions extracted from multiple categories so that the top-k suggestions[9] are treated by PQS as the most useful ones.

PQS combines the three computed scores of CS into a single measure using the *Stanford Certainty Factor* (SCF) [6], which allows multiple hypotheses to be considered in generating the ranking of CS belonged to c as

$$SCF(CS, c) = \frac{CL(CS, c) + NGS(CS, c) + Sim(CS, c)}{1 - Min\{CL(CS, c), NGS(CS, c), Sim(CS, c)\}} \tag{8}$$

Two candidate suggestions with the same sequence of terms can come from multiple categories. In the ranking process, only *one* instance is considered, i.e., the one with the highest $SCF(CS, c)$ value. The *higher* the SCF score of a candidate suggestion is, the *higher* its ranking is. By generating a single ranking score for CS using SCF, PQS selects the top ranked suggestions from different categories, which are *useful, diversified*, and *coherent* in terms of information represented by the corresponding terms.

H. Using a Prefix Trie to Correct Spelling Errors in Queries

Occasionally, a user *misspells* a word or forgets to add *spaces* in between keywords when posting a query Q. To enhance the *effectiveness* and *user-friendliness* of PQS, spelling errors are detected and corrected automatically using a *trie* data structure on words in an online dictionary or dictionaries, which avoids returning non-relevant or no results to the user.

During the process of parsing a user's query Q, PQS 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

[9] k in top-k suggestions is determined by the software developer who implements PQS and is recommended to be in the range of 5 and 10.

is encountered and no more characters are left in k, PQS 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. PQS 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 [11] 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*. PQS replaces the misspelled keyword in Q by the alternative one with the *highest* similarity percentage.

III. Experimental Results

In this section, we first present the dataset and performance metric employed for evaluating PQS (in Sections III-A and III-B, respectively). Thereafter, we analyze the results of the empirical study conducted to assess PQS and compare its performance with the ones achieved by existing commercial web search engines on query suggestions (in Section III-C).

A. The Dataset

To the best of our knowledge, there is no existing benchmark dataset that has been developed specifically for assessing the performance of query suggestion tools. For this reason, we constructed our own dataset, denoted QS-$Dataset$, which consists of 36 test queries to evaluate PQS. A gold standard is established for the suggestions of each test keyword query Q in QS-Dataset, which are determined by appraisers as ideal for Q.

With the majority of user queries submitted to existing web search engines including only one or two keywords,[10] we have considered only unigram and bigram test queries in our empirical study, instead of higher order n-gram queries that are seldom created by web search engine users. The 20 unigram and 16 bigram queries yield the representative set of test queries for assessing the suggestions made by PQS and other web search engines. Moreover, queries in QS-Dataset cover multiple topics and include keywords that can be interpreted differently by ordinary web search engine users to yield a diverse and unbiased set of test queries. (See Table II for the test queries used for evaluation purpose.)

To define the gold standard for the suggestions of each query in QS-Dataset, we conducted a user study using Amazon Mechanical Turk[11] which facilitates the task of compiling ideal suggestions on a given keyword query. The task was achieved by 143 Mechanical Turk appraisers who

[10]The AOL query log shows that 84% of user-submitted queries are either unigram or bigram queries.

[11]https://www.mturk.com/mturk/welcome

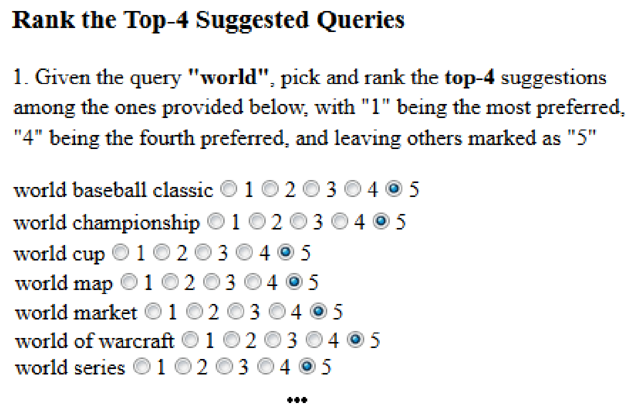
Table II
Test queries included in QS-Dataset

1. Baseball	13. Fish	25. National Security
2. Bee	14. Food Network	26. Native American
3. Biological	15. George	27. New York
4. British	16. Grand Slam	28. Presidential
5. Car Rental	17. Hakka Noodle	29. Rocket Launch
6. Chemical	18. Harry Potter	30. Soviet
7. Cherokee	19. Information	31. Space Shuttle
8. Chicken	20. Islamic	32. Vacation Rental
9. Communist	21. Kindergarten	33. Vegetarian Recipe
10. Computer Science	22. Launch	34. Weather
11. Database Management	23. Memorial	35. Wildlife
12. European Soccer	24. National Park	36. World

determined ideal suggestions for each query in QS-Dataset. We relied on Amazon's Mechanical Turk, since it is a "marketplace for work that requires human intelligence" that allows individuals or businesses to programmatically access thousands of diverse, on-demand workers [12] and has been used in the past to collect user feedback for multiple information retrieval tasks [13], [14].

We created a HIT (Human Intelligent Task) on Amazon's Mechanical Turk for each query Q in QS-Dataset. Given Q, each appraiser was presented with a set of up to 16 different suggestions for Q and asked to $select$ and $rank$ the top-4 that (s)he considered useful for Q. The (non-overlapped) suggestions on Q were previously generated by PQS, Google, Bing, and Yahoo! for comparison purpose. For any user-entered query on their respective web search engine, Bing and Yahoo! often provide between eight to ten suggestions, whereas Google offers only four. In order to conduct a fair performance evaluation, only the top-4 suggestions made by each of these web search engines and PQS were included in the corresponding HIT designed for Q, since four is the smallest number of suggestions (offered by Google among Bing, Yahoo!, and Google). In creating the HIT for Q, we set "5" to be the default value of each suggestion S to indicate that S is not a top-4 choice, which can be over-ruled by an appraiser who provided the top-4, i.e., from "1" to "4", ranked suggestions for Q. A sample HIT is shown in Figure 3.

The empirical study was conducted between June 6, 2013 and June 24, 2013 on Mechanical Turk. We collected 15 responses for each query Q in QS-Dataset. Based on the corresponding set of responses provided by Mechanical Turk appraisers, we defined a unique set of *four* ranked suggestions which yields the *gold standard* of Q using the *Borda Count voting scheme* [15]. The Borda Count voting scheme is a positional-scoring procedure such that given k ($\geq$ 1) candidates, each voter casts a vote for each candidate according to his/her preference. A candidate that is given a first-place vote receives k-1 points, a second-ranked candidate k-2 points, and so on up till the last candidate, who is awarded no points. Hereafter, the points assigned to each candidate across all the voters are added



Rank the Top-4 Suggested Queries

1. Given the query **"world"**, pick and rank the **top-4** suggestions among the ones provided below, with "1" being the most preferred, "4" being the fourth preferred, and leaving others marked as "5"

world baseball classic 1 2 3 4 [✓ 5]
world championship 1 2 3 4 [✓ 5]
world cup 1 2 3 4 [✓ 5]
world map 1 2 3 4 [✓ 5]
world market 1 2 3 4 [✓ 5]
world of warcraft 1 2 3 4 [✓ 5]
world series 1 2 3 4 [✓ 5]

•••

Figure 3. The HIT posted on Mechanical Turk to gather feedbacks on useful suggestions for the keyword query "world"

Table III
RANKINGS ASSIGNED TO SUGGESTIONS BY THE APPRAISERS FOR THE QUERY "WORLD", WHERE R STANDS FOR *Ranking* AND P FOR *Points*

Suggested Queries	Appraiser 1		. . .	Appraiser 15		Overall Points
	R	P		R	P	
world war	3	2	. . .	5	0	5
world championship	5	0	. . .	4	1	2
world cup	2	3	. . .	1	4	7
world of warcraft	5	0	. . .	2	3	3
world map	1	4	. . .	3	2	10
. . .	. . .	. . .	. . .	. . .	. . .	. . .

up and the candidate with the most points wins. PQS adopts Borda as an *aggregation strategy*, since its combination algorithm is simple and efficient, which requires neither training nor compatible relevance scores that may not be available. Furthermore, its performance is competitive with other existing combination strategies which typically require access to relevance scores. In the empirical study, we set k to be 5.

B. Performance Metric

We compute the *Normalized Discounted Cumulative Gain* (nDCG) [10] on the top-4 suggestions for each test query provided by PQS, Google, Bing, or Yahoo! to evaluate their performances. nDCG, as defined in Equation 9, determines the effectiveness of the query suggestion and ranking strategies of PQS, Google, Bing, and Yahoo!, respectively and $penalizes$ useful suggestions ranked $lower$ in a list of suggested queries. The penalization is based on a reduction, which is logarithmically proportional to the position of each useful query suggestion in a ranked list.

$$nDCG = \frac{1}{|QS-Dataset|} \sum_{i=1}^{|QS-Dataset|} \frac{DCG_i}{IDCG_i} \quad (9)$$

where $|QS\text{-}Dataset|$ is the total number of test queries in QS-Dataset, which is *thirty-six*, $IDCG_i$ (the Ideal Discounted Cumulative Gain) is the best possible DCG_i value for the ranked suggestions for the i^{th} test query[12], and

$$DCG_i = \sum_{j=1}^{4} \frac{2^{rel_j} - 1}{log_2(1+j)} \quad (10)$$

where rel_j is the binary relevant judgment of the top-j^{th} $(1 \leq j \leq 4)$ ranked suggestion and is assigned a value of "1" if the suggestion is *useful* and is "0", otherwise. A suggestion for a test query Q in QS-Dataset made by PQS (Google, Bing, or Yahoo!, respectively) is treated as *useful* if it is included in the *gold standard* for Q determined by Mechanical Turk appraisers as discussed in Section III-A

We have applied the *Wilcoxon signed-rank test*, which is a non-parametric test based on the differences between pairwise samples [10], to determine the statistical significance of the nDCG values achieved by PQS with respect to their counterparts obtained by Google, Bing, and Yahoo!, respectively using their suggestions for queries in QS-Dataset.

C. Performance Evaluation

We have quantified the effectiveness of PQS, Google, Bing, and Yahoo!, respectively in recommending useful queries and compared their performances based on the nDCG values computed on the thirty-six test queries in QS-Dataset as shown in Figure 4. When considering *unigram* queries, PQS achieved a statistical significance improvement in nDCG (as determined using a Wilcoxon signed-ranked test with $p < 0.001$) over Bing and Yahoo!, even though it is outperformed by a small, but statistically significant (with $p < 0.001$), margin by Google. In terms of making useful suggestions for *bigram* user queries, even though Bing and Yahoo! achieve a statistically significant difference on nDCG values (with $p < 0.001$ and $p < 0.005$, respectively) over PQS, PQS outperforms Google (with $p < 0.005$). By considering all the test queries in QS-Dataset and their results, PQS outperforms Yahoo! in suggesting useful queries, since the improvement in the overall nDCG value achieved by the former with respect to the latter is statistically significant (with $p < 0.001$), whereas Bing and Google achieve a very small, but statistically significant (with $p < 0.001$), difference in nDCG values over PQS.

Besides analyzing the overall performance of PQS, Google, Bing, and Yahoo!, we examined their performance at the *query level*. As shown in Figure 5, although Bing is able to suggest the perfect set of suggestions (i.e., top-4 suggestions included in the gold standard for the corresponding test query) for *sixteen* out of the 36 queries in QS-Dataset, it also provides *no* useful suggestions (according to Mechanical Turk appraisers) for *five* of the test queries. In reality, Google, Yahoo!, and PQS, generate a perfect set of suggestions for a *lower* number of queries (i.e., *eight*,

[12] $IDCG_i$ is computed using an *ideal ranking* on DCG_i such that the suggestions for the i^{th} test query in QS-Dataset are arranged in descending order based on their degrees of (non-)relevance to the test query.

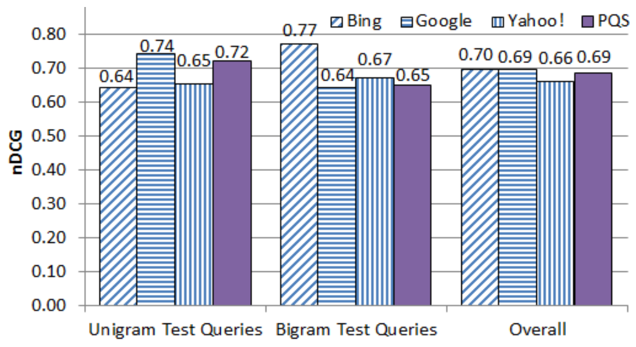


Figure 4. The nDCG scores for Bing, Google, Yahoo!, and PQS, respectively, which are determined using their suggestions for the test queries in QS-Dataset against the gold standards defined by appraisers

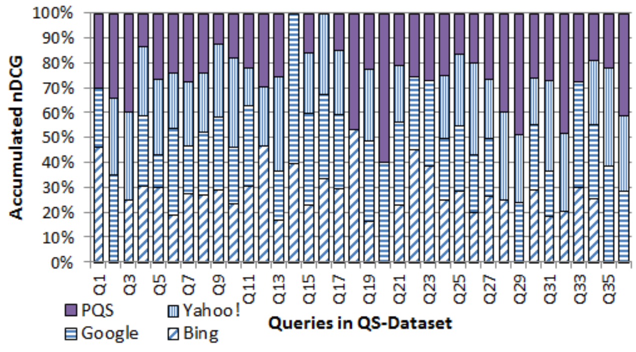


Figure 5. Per-query distribution of nDCG scores for Google, Bing, Yahoo!, and PQS, respectively

seven, and *seven*, respectively) than Bing. PQS, however, consistently suggests an average of close to *three* useful suggestions per query and does not suggest any useful queries for only *two* out of the 36 queries in QS-Dataset.

PQS also offers diverse suggestions which cover a wide range of information needs of distinct users, a task that cannot always be accomplished by its counterparts. As illustrated by the high nDCG value on Q_{20}, i.e., "Islamic", achieved by PQS as shown in Figure 5, PQS offers suggestions, which include "Islamic Law", "Islamic Philosophy," and "Islamic Theology," that capture varied search intentions that are appealing to different users, and thus are preferred by more appraisers involved in our empirical study, as opposed to suggestions provided by Bing (i.e., "Islamic Calendar" and "Islamic Calendar 2013") or Yahoo! (i.e., "Islamic Calendar" and "Islamic New Year") which capture the same search intent using alternative queries.

IV. Conclusions

In this paper, we have deviated from traditional query suggestion strategies that rely on large query logs by developing PQS, a probabilistic query suggestion module which analyzes text documents freely and publicly accessible online. PQS offers useful query suggestions to users that capture the *probability distribution*, *likelihood of co-occurrence*, and *cohesiveness* of keywords in suggestions across multiple pre-defined information categories. PQS is based on well-known, effective, and mathematically-sound probabilistic- and information retrieval-based models to make suggestions that address diverse users' search intents.

An empirical study conducted using a set of test queries and the gold-standards on suggestions for each test query established by using the feedbacks provided by Mechanical Turk appraisers has verified the effectiveness of the design and ranking strategy of PQS. The results of the study have demonstrated that (i) PQS outperforms Yahoo!, in terms of making useful query suggestions, and (ii) its performance is comparable to the ones achieved by Google and Bing.

References

[1] D. Kelly, K. Gyllstrom, and E. Bailey, "A Comparison of Query and Term Suggestion Features for Interactive Searching," in *ACM SIGIR*, 2009, pp. 371–378.

[2] Q. Mei, D. Zhou, and K. Church, "Query Suggestion Using Hitting Time," in *ACM CIKM*, 2008, pp. 469–478.

[3] X. Wang and C. Zhai, "Mining Term Association Patterns from Search Logs for Effective Query Reformulation," in *ACM CIKM*, 2008, pp. 479–488.

[4] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu, "Learning to Suggest: A Machine Learning Framework for Ranking Query Suggestions," in *SIGIR*, 2012, pp. 25–34.

[5] S. Bhatia, D. Majumdar, and P. Mitra, "Query Suggestions in the Absence of Query Logs," in *SIGIR*, 2011, pp. 795–804.

[6] G. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving,* 6^{th} *Ed.* Addison-Wesley, 2008.

[7] A. AramPatzis and J. Kamps, "A Study of Query Length," in *ACM SIGIR*, 2008, pp. 811–812.

[8] N. Phan, P. Bailey, and R. Wilkinson, "Understanding the Relationship of Information Need Specificity to Search Query Length," in *ACM SIGIR*, 2007, pp. 709–710.

[9] J. Koberstein and Y.-K. Ng, "Using Word Clusters to Detect Similar Web Documents," in *KSEM*, 2006, pp. 215–228.

[10] W. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2010.

[11] J. Oliver, "Decision Graphs - An Extension of Decision Trees," Monash University, Tech. Rep. 92/173, 1992.

[12] Amazon's Mechanical Turk, http://aws.amazon.com/mturk/.

[13] O. Alonso and M. Lease, "Crowdsourcing for Information Retrieval: Principles, Methods, and Applications," in *ACM SIGIR*, 2011, pp. 1299–1300.

[14] M. Koolen, J. Kamps, and G. Kazai, "Social Book Search: Comparing Topical Relevance Judgments and Book Suggestions for Evaluation," in *ACM CIKM*, 2012, pp. 185–194.

[15] J. Aslam and M. Montague, "Models for Metasearch," in *ACM SIGIR*, 2001, pp. 276–284.