Help Me Search: Leveraging User-System Collaboration for Query Construction to Improve Accuracy for Difficult Queries

Saar Kuzi
skuzi2@illinois.edu
University of Illinois at Urbana-Champaign

Abhishek Narwekar*
narweka@amazon.com
Amazon Alexa

Anusri Pampari*
anusri@stanford.edu
Stanford University

ChengXiang Zhai
czhai@illinois.edu
University of Illinois at Urbana-Champaign

ABSTRACT

In this paper, we address the problem of difficult queries by using a novel strategy of collaborative query construction where the search engine would actively engage users in an iterative process to continuously revise a query. This approach can be implemented in any search engine to provide search support for users via a "Help Me Search" button, which a user can click on as needed. We focus on studying a specific collaboration strategy where the search engine and the user work together to iteratively expand a query. We propose a possible implementation for this strategy in which the system generates candidate terms by utilizing the history of interactions of the user with the system. Evaluation using a simulated user study shows the great promise of the proposed approach. We also perform a case study with three real users which further illustrates the potential effectiveness of the approach.

ACM Reference format:
Saar Kuzi, Abhishek Narwekar, Anusri Pampari, and ChengXiang Zhai. 2019. Help Me Search: Leveraging User-System Collaboration for Query Construction to Improve Accuracy for Difficult Queries. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, July 21–25, 2019 (SIGIR '19),* 4 pages.
https://doi.org/10.1145/3331184.3331362

1 INTRODUCTION

The current search engines generally work well for popular queries where a large amount of click-through information can be leveraged. Such a strategy may fail for long-tail queries, which are entered by only a small number of users. Thus, for such queries, a search engine generally would have to rely mainly on matching the keywords in the query with those in documents. Unfortunately, such a method would not work well when the user's query does not include the "right" keywords. Users in such cases would often end up repeatedly reformulating a query, yet they still could not find the relevant documents. Unfortunately, there are many such queries, making it a pressing challenge for search engines to improve their accuracy.

In this paper, we address this problem and propose a general strategy of collaborative query construction where the search engine would actively engage users in an iterative process to revise a query. The proposed strategy attempts to optimize the collaboration between the user and the search engine and is based on the following assumptions: (1) *Ideal query*: For any difficult query, there exists an ideal query that, if constructed, would work well. This assumption allows us to re-frame the problem of how to help users as the problem of how to construct an ideal query. (2) *User-system collaboration*: User-system collaboration can be optimized by leveraging the strength of a search engine in "knowing" all the content in the collection and the strength of a user in recognizing a useful modification for the query among a set of candidates. (3) *User effort*: When facing a difficult query, the user would be willing to make some extra effort to collaborate with the search engine.

Our main idea is to optimize the user-system collaboration in order to perform a sequence of modifications to the query with the goal of reaching an ideal query. While the proposed strategy includes multiple ways to edit the query, we initially focus on studying a specific editing operator where the system suggests terms to the user to be added to the query at each step based on the history of interactions of the user with the system.

We perform an evaluation with a simulated user which demonstrates the great promise of this novel collaborative search support strategy for improving the accuracy of difficult queries with minimum effort from the user. The results also show that suggesting terms based on user interaction history improves effectiveness without incurring additional user effort. Finally, we conduct a case study with three real users that demonstrates the potential effectiveness of our approach when real users are involved.

2 RELATED WORK

The main novelty of our work is the idea of collaborative construction of an ideal query, specific algorithms for iterative query expansion, and the study of their effectiveness for difficult queries.

Previous works have studied approaches for interactive query expansion (e.g., [2, 4, 10]). According to these works, the user needs to select terms to be added to each query independently. Our framework is more general both in performing a sequence of query modifications to optimize the user-system collaboration and in allowing potentially other query modifications than simply adding terms.

*This work was done while the author was a student at UIUC.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331362

Furthermore, we propose methods which suggest terms to the user based on the history of interactions of the user with the system.

On the surface, our approach is similar to query suggestion already studied in previous works [3]. However, there are two important differences: (1) The suggested queries in our approach are expected to form a sequence of queries incrementally converging to an ideal query whereas query suggestion is done for each query independently. (2) The suggested queries in our method are composed of new terms extracted from the text collection, but the current methods for query suggestion tend to be able to only suggest queries taken from a search log.

Other works focused on developing query suggestion approaches for difficult queries [6, 11]. In general, ideas from past works on query suggestion can be used in our approach for generating the set of query modifications that are suggested to the user.

There is a large body of work on devising approaches for automatic query reformulation. One common method is to automatically add terms to the user's query [5]. Other approaches include, for example, substitution or deletion of terms [12]. The various ideas, which are suggested by works in this direction, can be integrated into our collaborative approach by devising sophisticated methods for term suggestion.

3 COLLABORATIVE QUERY CONSTRUCTION

Our suggested Collaborative Query Construction (CQC) approach is based on the **Ideal Query Hypothesis (IQH)**, which states that for any information need of a user, there exists an ideal query that would allow a retrieval system to rank all the relevant documents above the non-relevant ones. The IQH implies that if a user has perfect knowledge about the document collection, then the user would be able to formulate an ideal query. The IQH is reasonable because it is generally possible to uniquely identify a document by just using a few terms that occur together in it but not in others. This point was also referred to in previous work as the *perfect query paradox* [8]. We note that the IQH may not always hold; for example, when there are duplicate documents. Nevertheless, it provides a sound conceptual basis for designing algorithms for supporting users in interactive search. Based on the IQH, the problem of optimizing retrieval accuracy can be reduced to the problem of finding the ideal query. Thus, based on this formulation, the main reason why a search task is difficult is that the user does not have enough knowledge to formulate the ideal query. In this paper, we address this problem by helping a user to construct an ideal query.

Our collaborative query construction process is represented by a sequence of queries, $Q_1, Q_2, ..., Q_n$, where Q_1 is the user's initial query, Q_n is an ideal query, and Q_{i+1} is closer to Q_n than Q_i and the gap between Q_i and Q_{i+1} is small enough for the user to recognize the improvement of Q_{i+1} over Q_i. From the system's perspective, at any point of this process, the task is to suggest a set of candidate queries, while the user's task is to choose one of them. In this paper, we focus on a specific approach in which the query refinement is restricted to only adding one extra term to the query at each step. That is, a single collaborative iteration of revising a query Q_i would be as follows: (1) Present the user a list of m candidate terms, T_i (not already selected). (2) The user selects a term, $t \in T_i$. (3) $Q_{i+1} = Q_i \bigcup \{t\}$. (4) Q_{i+1} is used to retrieve a result list D_{i+1}.

One advantage of using such an approach is that the gap between two adjacent queries is expected to be small enough for the user to recognize the correct choice. Furthermore, although this implementation strategy is very simple, theoretically speaking, the process can guarantee the construction of any ideal query that contains all the original query terms if the system can suggest additional terms in the ideal query but not in the original query and the user can recognize the terms to be included in the ideal query. We assume that the original query terms are all "essential" and should all be included in the ideal query. While true in general, in some cases this assumption may not hold, which would require the removal or substitution of terms in the initial query. In this paper, however, we focus on term addition as our first strategy and leave the incorporation of other operations for future work.

Following the game-theoretic framework for interactive IR [13], our approach can be framed as the following Bayesian decision problem where the goal is to decide a candidate set of terms T_i to suggest to the user in response to the current query Q_i:

$$T_i = \arg \min_{T \subset V - Q_i} \int_{\Theta_Q} L(T, H_i, \Theta_Q, U) p(\Theta_Q | H_i, U) d\Theta_Q; \quad (1)$$

where (1) T_i is a candidate set of terms to be presented to the user (a subset of the vocabulary V). (2) H_i is all the information from the history of interactions of the user with the system. (3) Θ_Q is a unigram language model representing a potential ideal query. (4) U denotes any relevant information about the user. (5) $L(T, H_i, \Theta_Q, U)$ is a loss function assessing whether T is a good choice for H_i, U, and Θ_Q. (6) $p(\Theta_Q|H_i, U)$ encodes the current belief about the ideal query. The integral indicates the uncertainty about the ideal query, which can be expected to be reduced as we collect more information from the user.

While in general we need to assess the loss of an entire candidate set T, in the much simplified method that we will actually explore, we choose T by scoring each term and then applying a threshold to control the number of terms. That is, we assume that the loss function on a term set T can be written as an aggregation of the loss on each individual term. As an additional simplification, we approximate the integral with the mode of the posterior probability about the ideal query, $\hat{\Theta}_Q$. Thus, our decision problem would become to compute the score of each term t, not already selected by the user, as follows: $s(t) = -L(t, H_i, \hat{\Theta}_Q, U)$; where $\hat{\Theta}_Q = \arg\max_{\Theta_Q} p(\Theta_Q|H_i, U)$. Computationally, the algorithm boils down to the following two steps: (1) Given all of the observed information H_i and U, compute $\hat{\Theta}_Q$. (2) Use $\hat{\Theta}_Q$ along with H_i and U to score each term in the vocabulary but not already in Q_i.

4 TERM SCORING

According to the previous section, the optimal scoring function $s(t)$ is based on the negative loss $-L(t, H_i, \hat{\Theta}_Q, U)$. Intuitively, the loss of word t is negatively correlated with its probability according to $\hat{\Theta}_Q$. We thus simply define our scoring function as $s(t) = p(t|\hat{\Theta}_Q)$. That is, our problem is now reduced to infer $\hat{\Theta}_Q$ given all of the observed information H_i and U.

Next, we suggest a model for inferring $\hat{\Theta}_Q$, which is based on Pseudo-Relevance Feedback (PRF). This model is an extension of the relevance model RM1 [7] to incorporate H_i (We leave the incorporation of U for future work as such data is not available to us.):

$$p(t|\hat{\Theta}_Q) = \sum_{d \in D_i} p(t|d) \cdot p(d|Q_1, H_i). \quad (2)$$

$p(t|d)$ is estimated using the maximum likelihood approach. We approximate $p(d|Q_1, H_i)$ using a linear interpolation:

$$p(d|Q_1, H_i) = (1-\alpha) \cdot p(d|Q_1) + \alpha \cdot p(d|H_i); \quad (3)$$

$p(d|Q_1)$ is proportional to the reciprocal rank of d w.r.t Q_1; $\alpha \in [0, 1]$. In order to estimate $p(d|H_i)$, two types of historical information are considered: (1) The terms selected by the user previously (H_i^T). (2) The result lists presented to the user previously (H_i^D). We combine these two components as follows: $p(d|H_i) = p(d|H_i^D) \cdot p(H_i^D|H_i) + p(d|H_i^T) \cdot p(H_i^T|H_i)$. (We assume $p(H_i^D|H_i) = p(H_i^T|H_i)$.)

In order to estimate $p(d|H_i^D)$, we assume that documents which appear in the result list presented to the user in the current iteration, and that were absent in the previous result list, represent aspects of the information need that are more important to the user. We thus estimate $p(d|H_i^D)$ as follows:

$$p(d|H_i^D) = \frac{1}{rank_{D_i}(d) \cdot Z_D} \; \forall d \in D_i \setminus D_{i-1}; \quad (4)$$

$p(d|H_i^D) = 0$ for all other documents; $rank_{D_i}(d)$ is the rank of document d in the result list D_i; Z_D is a normalization factor.

We estimate $p(d|H_i^T)$ such that high importance is attributed to documents in which terms that were previously selected by the user are prevalent.

$$p(d|H_i^T) = \sum_{j=1}^{i-1} p(d|t_j, H_i^T) \cdot p(t_j|H_i^T); \quad (5)$$

t_j is the term selected by the user in the j'th iteration. $p(d|t_j, H_i^T)$ is set to be proportional to the score of d with respect to t_j as calculated by the system's ranking method. Assuming that terms selected in more recent iterations are more important than older ones, we estimate $p(t_j|H_i^T)$ as: $p(t_j|H_i^T) = \frac{\exp(-\mu \cdot (i-j))}{Z_T}$; Z_T is a normalization factor; μ is a free parameter and is set to 0.5.

To conclude, we assign a probability to each term which is a linear interpolation of its probabilities in the documents in the result list, where the interpolation weights are influenced by: (1) the rank of the document, (2) the presence of the document in the previous list, and (3) the frequency of terms that were previously selected.

Query representation: According to our approach, the query Q_i is composed of the original query Q_1 and the terms selected by the user. The terms in Q_i are weighted based on a probability distribution such that the probability of a term t in V is: $p(t|Q_i) = \lambda_i \cdot p_{mle}(t|Q_1) + (1-\lambda_i) \cdot p(t|H)$; $p(t|H)$ is proportional to the weight that was assigned to the term by the scoring method if this term was previously selected, and is set to 0 otherwise; $p_{mle}(t|Q_1)$ is the maximum likelihood estimate of t in Q_1; $\lambda_i \in [0, 1]$.

5 EVALUATION

The evaluation of the proposed strategy has two challenges: (1) The proposed approach is of interactive nature. (2) We are interested in focusing on difficult queries. We address these challenges by constructing a new test collection based on an existing collection that would focus on difficult queries and experimenting with simulated users. Finally, we perform a case study with three real users.

Experimental setup: We use the ROBUST document collection (TREC discs 4 and 5-{*CR*}). The collection is composed of 528,155 newswire documents, along with 249 TREC topics which their titles serve as queries (301-450, 601-700). Stopword removal and Krovetz stemming were applied to both documents and queries. The Lucene toolkit was used for experiments (lucene.apache.org). The BM25 model was used for ranking [9]. We use the following strategy to construct our test set. We first perform retrieval for all queries. Then, we remove from the collection the relevant documents that are among the top 10 documents in each result list. After doing that, we remain with 105 queries for which $p@10 = 0$ when performing retrieval over the modified collection. We use these queries for our evaluation, along with the modified collection. We report performance in terms of precision ($p@\{5, 10\}$) and Mean Reciprocal Rank (*MRR*@1000), which is more meaningful than Mean Average Precision in the case of such difficult queries (it measures how much effort a user has to make in order to reach the very first relevant document). We also report the fraction of queries for which a method resulted in $p@10 > 0$, denoted *success*@10. The two-tailed paired t-test at 95% confidence level is used in order to determine significant differences in performance.

Table 1: Simulated user performance. Statistically significant differences with RM3 are marked with asterisk. All differences with the initial query are statistically significant.

	Single Term				Five Terms			
	p@5	*p*@10	*MRR*	*success*@10	*p*@5	*p*@10	*MRR*	*success*@10
Initial	.000	.000	.053	.000	.000	.000	.053	.000
RM3	.036	.040	.083	.238	.040	.049	.090	.219
CQC	.057	.090*	.127*	.457	.137*	.136*	.209*	.447

Our approach involves free parameters, which are set to effective ones following some preliminary experiments. We should point out that our research questions are mainly about how promising the proposed approach is as a novel interaction strategy, which is generally orthogonal to the optimization of these parameters. The number of terms suggested to the user, m, is set to 5. The number of documents used in our PRF-based term scoring method is set to 100. The interpolation parameter in Equation 3, α, is set to 0.8. The value of λ_i, the weight given to the original query, is set to $\max(0.4, \frac{|Q_1|}{|Q_i|})$; we chose this weighting function as to attribute high importance to the original query when a small amount of expansion is used. We compare the performance of our approach with that of using the original query, and of using an automatic query expansion approach in which a set of terms is automatically added to the original query once. We set the number of expansion terms to be equal to the number of terms that were added by the user in the collaborative process. We use the RM3 [1] expansion model (free parameters are set as in the collaborative approach).

Simulation study: In order to do a controlled study of our approach, we experiment with a simulated user. Given a list of term suggestions, the simulated user chooses a term with the highest $tf.idf$ score in the relevant documents. Specifically, for each query we concatenate all relevant documents and compute $tf.idf$ based on the single concatenated "relevant document". Our main result for the simulated user experiment is presented in Table 1. We report the performance when a single term or five terms are added. According to the results, the collaborative approach (CQC) is very effective. Specifically, after adding a *single* term to the query, users are able to

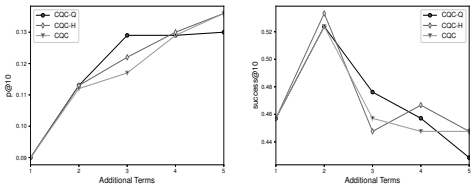

Figure 1: Performance of the different model components.

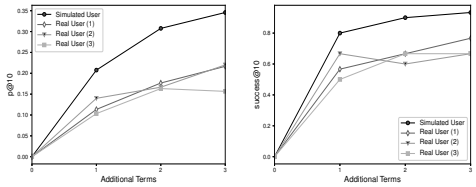

Figure 2: Real users vs. Simulated user.

see a noticeable improvement on the first page of search results in about 45% of these difficult queries that did not return any relevant document initially (*success*@10). Furthermore, our approach outperforms the initial query to a statistically significant degree for all evaluation measures (for both number of terms) and is also much more effective than RM3. Our term scoring method utilizes both the original query and the user interaction history. We are interested in examining the relative importance of these individual components. Setting $\alpha = 0$ in Equation 3 results in a model that uses only the original query (CQC-Q). Setting $\alpha = 1$ results in a model that uses only user history (CQC-H). The results are presented in Figure 1. Focusing on p@10, we can see that all components are very effective. Comparing the different components, we can see that CQC-H is outperformed by CQC-Q for a small number of terms, and the opposite holds for a large number. In terms of *success*@10, we can see that all model components achieve the highest performance when two terms are added, with CQC-H being the best performing one. Interestingly, *success*@10 decreases as more terms are added. That is, while adding more terms to the query can improve the average performance, it results in a less robust approach.

Case study with real users: We are interested in examining whether real users can recognize the "good" terms suggested by the system. To gain some initial understanding regarding this issue, we conducted a case study with three real users. We note that the conclusions that can be drawn from this study are limited due the small number of users. Yet, this study is still useful for getting some intuition regarding the utility of the approach. Each participant performed three iterations of the collaborative process for 30 queries. Specifically, we selected queries that achieved the highest performance in terms of p@10 after adding a single term by the simulated user. We chose these queries as we are interested to study the following research question: given a term scoring method that can provide effective terms, can the user identify them? For each query, the user was presented with the initial query, a text describing the topic, and the guidelines regarding how a relevant document should look like (all are part of the TREC topics). After issuing a query, the users are presented with a result list of 10 documents (a title and a short summary of 5 sentences are presented).

In Figure 2, we compare the performance of the real users with that of the simulated user. According to the results, retrieval performance can be very good when terms are selected by real users. Specifically, all users reach *success*@10 of around 0.5. That is, after adding a single term, at least one relevant result is obtained for about 50% of the queries. In Table 2, we present examples of queries along with the terms that were selected by a single real user and a simulated user. We also report the performance that resulted from adding a term. The first query serves as an example where the real user outperforms the simulated user by a better choice of terms. The second query is an example where the simulated user outperforms the real user presumably by recognizing the correct technical terms. Finally, the last query is an example where both users achieve similar performance, but using different terms.

Table 2: Query Examples. The performance of the query (p@10) after adding a term is reported in the brackets.

curbing population growth	Real User	plan (0.0)	family (0.2)	birth (0.6)
	Simulated	china (0.1)	economic (0.1)	rate (0.2)
Stirling engine	Real User	company (0.0)	financial (0.0)	group (0.0)
	Simulated	cfc (0.9)	hcfc (1.0)	hyph (1.0)
antibiotics ineffectiveness	Real User	infection (0.2)	research (0.2)	study (0.2)
	Simulated	drug (0.1)	pharmaceutical (0.2)	product (0.1)

6 CONCLUSIONS AND FUTURE WORK

We proposed and studied a novel strategy for improving the accuracy of difficult queries by having the search engine and the user collaboratively expand the original query. Evaluation with simulated users and a case study with real users show the great promise of this strategy. In future work, we plan to devise more methods for term scoring, incorporate more operations for query modification, and perform a large-scale user study.

Acknowledgments. This material is based upon work supported by the National Science Foundation under grant number 1801652.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. [n. d.]. *UMass at TREC 2004: Novelty and HARD.* Technical Report.
[2] Peter Anick. 2003. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of SIGIR*. ACM, 88–95.
[3] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *International Conference on Extending Database Technology*. Springer, 588–596.
[4] Nicholas J. Belkin, Colleen Cool, Diane Kelly, S-J Lin, SY Park, J Perez-Carballo, and C Sikora. 2001. Iterative exploration, design and evaluation of support for query reformulation in interactive information retrieval. *Information Processing & Management* 37, 3 (2001), 403–434.
[5] Claudio Carpineto and Giovanni Romano. 2012. A Survey of Automatic Query Expansion in Information Retrieval. *ACM Comput. Surv.* (2012).
[6] Van Dang and Bruce W Croft. 2010. Query reformulation using anchor text. In *Proceedings of WSDM*. ACM, 41–50.
[7] Victor Lavrenko and W Bruce Croft. 2001. Relevance based language models. In *Proceedings of SIGIR*. ACM, 120–127.
[8] David Dolan Lewis. 1992. *Representation and learning in information retrieval.* Ph.D. Dissertation. University of Massachusetts at Amherst.
[9] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR*. Springer-Verlag New York, Inc., 232–241.
[10] Ian Ruthven. 2003. Re-examining the potential effectiveness of interactive query expansion. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 213–220.
[11] Yang Song and Li-wei He. 2010. Optimal rare query suggestion with implicit user feedback. In *Proceedings of WWW*. ACM, 901–910.
[12] Xuanhui Wang and ChengXiang Zhai. 2008. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of CIKM*. ACM, 479–488.
[13] ChengXiang Zhai. 2016. Towards a game-theoretic framework for text data retrieval. *IEEE Data Eng. Bull.* 39, 3 (2016), 51–62.