Query formulation with a search assistant

Fu, Lin; Goh, Dion Hoe-Lian; Supangat, Yohan; Foo, Schubert

2004

Fu, L., Goh, D., Foo, S., & Supangat, Y. (2004). Query formulation with a search assistant. Proceedings of the 7th International Conference on Asian Digital Libraries ICADL 2004, (December 13-17, Shanghai, China), Lecture Notes in Computer Science 3334, 491-500.

<https://hdl.handle.net/10356/91184>

<https://doi.org/10.1007/b104284>

The original publication is available at www.springerlink.com.

Downloaded on 26 Apr 2023 23:12:51 SGT

Fu, L., Goh, D.H., Foo, S., and Supangat, Y. (2004). Query formulation with a search assistant. *Proceedings of the 7th International Conference on Asian Digital Libraries ICADL 2004*, (December 13-17, Shanghai, China), *Lecture Notes in Computer Science 3334*, 491-500.

Query Formulation with a Search Assistant

Lin Fu, Dion Hoe-Lian Goh, Schubert Shou-Boon Foo, Yohan Supangat

Division of Information Studies
School of Communication and Information
Nanyang Technological University
Singapore 637718

{p148934363, ashlgoth, assfoo}@ntu.edu.sg, fyohans@pmail.ntu.edu.sg

Abstract. Information overload has led to a situation where users are swamped with too much information, resulting in difficulty sifting through the material in search of relevant content. In this paper, we address this issue from the perspective of collaboration in query formulation. We describe a search assistant that helps users with query formulation by finding related previously submitted queries through mining query logs. The search assistant runs as a reusable software component and can be incorporated into various search engines. We report our approach to designing and implementing the software and evaluation results.

1 Introduction

With the proliferation of online search engines, more attention has been paid to assist the user in formulating an accurate query to express his/her information needs. A number of approaches have been proposed. One approach is to use interactive query reformulation systems which aim to detect a user's "interests" through his/her submitted queries and give users opportunities to rephrase their queries by suggesting alternate queries.

There are two approaches to obtaining the recommended query terms: (1) through analyzing the results returned by the search engine corresponding to the initial query and using the most frequently occurring terms as recommendations [2, 3]; (2) through mining query logs and using similar queries issued by other users as recommendations, which is also known as collaborative querying [4, 8]. A common technique in collaborative querying is known as query clustering, which groups similar queries automatically without using predetermined class descriptions. A query clustering algorithm could provide a list of suggestions by offering, in response to a query Q , the other members of the cluster containing Q . In this way, there is an opportunity for a user to take advantage of previous queries and use the appropriate ones to meet his/her information need.

Since similarity is fundamental to the definition of a cluster, measures of similarity between two queries are essential to the query clustering procedure. We have developed a hybrid query similarity measure that exploits both the query terms and query results URLs. Experiments reveal that using the hybrid approach, more balanced query clusters can be generated than using other techniques. We have developed a search assistant which exploits the hybrid similarity measure to cluster

queries and a graph visualization approach to represent the query clusters. The system gives users the opportunity to rephrase their queries by suggesting alternate queries.

The rest of this paper is organized as follows. In Section 2, we review the literature related to this work. We then present the design and implementation of the search assistant, the Query Graph Visualizer (QGV). Next, we report a user study to assess the usefulness and usability of the QGV. Finally, we discuss the implications of our findings for collaborative querying systems and outline areas for future work.

2 Related Work

There are several useful strands of literature that bear some relevance to this work. This section briefly reviews literature from these fields. Firstly, a survey of interactive query reformulation is provided as the background for this research. Next, a review of different query clustering approaches is presented.

Several techniques have been used to incorporate aspects of interactive query reformulation systems into the information retrieval process. One technique to obtain similar queries is to use terms extracted from the search results documents. Examples include HiB [3], Paraphrase [1] and Altavista Prisma [2], which parse the list of result documents and use the most frequently occurring terms as recommendations. Another approach is collaborative querying. Related queries (the query clusters) may be calculated based on the similarities of the queries in the query logs [9] which provide a wealth of information about past search experiences. The system can then either recommend the similar queries to users [9] or use them as expansion term candidates to the original query to augment the quality of the search results [5]. Here, calculating the similarity between different queries and clustering them automatically are crucial steps.

Traditional information retrieval research suggests an approach to query clustering by comparing query term vectors (content-based approach) [12]. Raghavan and Sever [11] determine similarity between queries by calculating the overlap in documents returned by the queries (results-based approach). Fitzpatrick and Dent [7] further developed this method by weighting the query results according to their position in the search results list. Glance [9] uses the overlap of result URLs as the similarity measure instead of the document content.

3 Query Graph Visualizer: A Search Assistant

We have designed a search assistant, the Query Graph Visualizer (QGV) based on the query clusters generated using a hybrid query similarity measure [8]. In our system, the query clusters can be explored using a graph visualization scheme.

3.1 System Architecture

Figure 1 sketches the architecture of the QGV. After capturing a new query, the system will identify related queries and use them as recommended queries to users. The recommended queries are displayed in HTML format, similar to [1, 2, 5]. Users may further explore the recommended queries by visualizing the query clusters as a graph which contains the initial query and recommended queries. The QGV is designed to be an independent agent and can be incorporated into different information retrieval systems.

Fig. 1. Architecture of the collaborative querying system

It can be seen from the architecture that there are three essential processes to accomplish collaborative querying. The first is the query repository construction procedure which involves query cluster generation. The second process is the maintenance of the query repository. The third process is the query recommendation phase that includes related query detection and query graph visualization.

3.2 Query Repository Construction

As mentioned previously, the content-based approach uses overlap between query terms as the similarity measure to cluster the queries. However the content-based approach might not be appropriate for query clustering since most queries submitted to search engines are quite short [13, 14]. Thus query terms can neither convey much information nor help to detect the semantics behind them since the same term might represent different semantic meanings, while on the other hand, different terms might refer to the same semantic meaning. For the results-based approach, the same document in the search results listings might contain several topics, and thus queries with different semantic meanings might lead to the same search results. Thus, we hypothesize that using both query terms and the corresponding results may compensate for the drawbacks inherent in each method. The hybrid approach is a linear combination of the content-based and results-based approaches [8]. The effect of each individual approach on the hybrid approach can be controlled by assigning a weight parameter to them respectively. Here, two queries are similar when (1) they contain one or more terms in common; or (2) they have results that contain one or

more items in common. Further, two queries are in one cluster whenever their similarity is above a certain threshold.

The steps to construct the query repository include: (1) preprocessing the queries; (2) retrieving result URLs from the search engine and calculating the similarity between queries based on the URLs; (3) computing query term weights according to the TFIDF method and calculating the similarity between queries based on the cosine similarity measure; (4) combining the content-based and result-based similarity values to generate the hybrid similarity between queries; (5) generating the query clusters.

Our previous experiments confirm our hypothesis that a combination of both query terms and result URLs provide a better overall quality of query clusters than using each separately. More information about the hybrid approach can be found in [8].

3.3 Query Repository Maintenance

When new queries arrive at the system, there is a need to update the query repository so that the system can harness and recommend the latest useful queries. This process can be done periodically offline. Since these new queries will affect the weights of the existing query terms, all these queries will be captured. If a new query already exists in the query database, then we simply update the corresponding query weights (TFIDF) and similarity values. For those that are unique, the new queries incorporated into the query database similar to the query repository construction process (see Section 3.2).

3.4 Query Recommendation

This process involves identifying related queries in the query repository constructed in the previous step and visualizing the related queries.

3.4.1 Detecting Related Queries

Given a query submitted by a user, we search the query repository for related queries and then recommend these to the user. Here, a recursive algorithm is implemented to search for query clusters in the query repository with the initial query regarded as the root node. First, the system will detect the query cluster $G(Q_i)$ containing the initial query Q_i . Given the definition of a query cluster (Section 3.2), all its members are directly related to Q_i . Therefore $G(Q_i)$ can be regarded as the first level in the graph structure of all queries related to Q_i , as shown in Figure 2. Besides the query cluster $G(Q_i)$, the system will further find query clusters containing the members of $G(Q_i)$. For example since Q_1 is a member in the cluster $G(Q_i)$, the system will locate the query cluster $G(Q_1)$ which forms the second level of the graph structure as shown in Figure 2. This recursive process will stop at the user-specified maximum level to be searched. In our algorithm, the default maximum value is 5 which means the algorithm will only detect the top five levels of query clusters related to the root node. Thus the final set of related queries to Q_i might go beyond the members within $G(Q_i)$, giving a range of recommended queries directly or indirectly related to Q_i . If a query is not found in the repository, the system will split the query terms and provide statistical information on each individual term. For example, for the query “data mining”, the QGV will tell the user how many queries are related to the term “data”

and the term “mining”. At the same time, the query repository will be updated with this new query.

Fig. 2. Detecting related queries

3.4.2 Query Cluster Visualization

Our system, the QGV, displays query clusters in a graph (Figure 3). The graph edges show the relationship between two graph nodes, with the value on the edge indicating the strength of the relationship. For example, 0.1 on the edge between the nodes “data mining” and “predictive data mining” indicates the similarity weight between these two nodes is 0.1. In addition, the system offers a control tool bar to manipulate the graph visualization area including zooming, rotating and locality zooming. The zooming function allows users to shrink or enlarge the graph visualization area. The rotating function allows users to view the visualization area from different directions. Finally, locality zooming refers to the number of levels of related queries to be displayed.

By right clicking on an individual node, a popup menu appears, offering a variety of options. Firstly, users can use the selected query node and post it to a search engine (e.g. the Digital Library at Nanyang Technological University, Google, etc.). Recall that the QGV is running as an independent agent and can be incorporated into various search engines. Secondly, users may use this query to carry out a new search across the query repository and locate queries related to the selected one. Further, users can expand and collapse each query node on the graph. The QGV is implemented using “Touchgraph” which is an open source component to visualize information in graph formats (<http://touchgraph.sourceforge.net>).

Fig. 3. Query graph visualizer

4 Evaluation

An evaluation was conducted on the QGV through a questionnaire in a controlled environment to assess its usefulness and usability. Here, we adopt Nielsen's heuristic evaluation approach [10]. This technique is used to find usability problems by getting a small number of evaluators to examine an interface and judge its compliance with ten recognized usability principles. The goal is to obtain the most useful information for guiding re-design with the least cost. In this section, we report our evaluation design and results.

4.1 Evaluation Design

Twelve participants were involved in this evaluation. Among the twelve, seven participants have a Bachelor's degree and five subjects have a Master's degree. Seven participants had a computer science background, four were from various areas in engineering and one was from communication studies. All of participants used computers everyday and said they had good skills in using online search engines.

The twelve participants were randomly divided into two groups of six participants each. Group A used the QGV to finish two search tasks while group B used the Google search engine to finish the same tasks as group A. Participants in group A were given a ten minute introduction to the QGV followed by a practice session before carrying out the tasks. Participants in group B were introduced to the QGV after they finished the tasks using Google and asked to try the system. The first task was to “find information on a semantic web seminar held in 2002 which discussed and explored various rule system techniques that are suitable for the web”. The second task was to “find information about a web project which aims to represent mathematical documents in a form understandable by humans and interpretable by machines on World Wide Web”. The time taken to accomplish the tasks was recorded. At the end of the evaluation, the participants were asked to complete a preference questionnaire on the QGV.

4.3 Evaluation Results

Table 1 shows a noticeable difference between groups in the time required to complete the tasks accurately. Participants in group A exhibited a major reduction in terms of average time to complete the tasks. This suggests that the QGV helped the participants find the desired information more quickly than Google alone. In other words, it appears that the QGV can help users formulate better queries by harnessing other information seekers’ knowledge and reduce the time needed to sift through search result documents in search of relevant content.

	Group A	Group B
Task 1	3 min	13.6 min
Task 2	3 min	23.4 min
All tasks	6 min	37 min

Table 1. Average Time to Complete Tasks for Each Group

With regard to the usability issue, we evaluated the QGV in terms of Nielsen’s 10 heuristic principles. In our evaluation, each heuristic principle is reflected by one or more questions concerning the design of the QGV. Each question is rated along a five-point scale – “strongly disagree”, “disagree”, “neutral”, “agree” and “strong agree”. Due to space limitations, we only report the number of participants who rank “agree” or “strongly agree” (see Table 2). As shown, the QGV satisfies most of ten heuristic principles according to our participants. For example, in “visibility of system status”, 10 participants agreed that the status bar at the bottom was informative enough to reflect the action status of the system. As far as “consistency and standards” is concerned, 10 participants were comfortable with the graphic and layout design in the QGV and agreed that the font size, color, buttons, text box and popup menu were consistent; etc. The results thus indicate that the QGV performs well in terms of usability issues.

Heuristic principles	Corresponding questions	Number of participants who rank “agree” or “strongly agree”
Visibility of system status	The status bar at the bottom has enough information to reflect the action status of the system.	10
Match between system and real world	The language used in the system is phrased in Standard English instead of technical terms.	10
	The language used in the system is easy to understand.	10
User control and freedom	The zoom graph menu function under the graph navigation option is easy to use.	12
	The rotate graph menu function under the graph navigation option is easy to use.	8
	The locality graph menu function under the graph navigation option is easy to understand and useful in reducing the complexity of graphical display.	6
	Overall, it is easy and convenient to navigate within the query graph.	10
Consistency and standards	The font size, type, color, buttons, text box and dropdown box menu are consistent.	10
	There are informative tool tips on each button, text box and dropdown box control menu.	12
Error prevention	The system prompt when a searched query list cannot be found is useful.	8
Recognition rather than recall	The graph option dropdown box containing zoom, rotate and locality messages are easily understood	8
	The Query Node click and drag feature is noticeable.	10
	The Query Node popup menu that appears when right-clicking is noticeable.	10
	The use of gradation color to differentiate query node level and edge weight helps the user understand the relatedness between query nodes better.	10
Flexibility and efficiency of use	In overall, the Query Graph Visualization system speeds up the search process compares to other conventional search engines, eg. Google.	12
Aesthetic and minimalist design	The screen elements are not cluttered and placed appropriately on the screen.	10
	The dropdown boxes, buttons and text labels provide adequate information to perform the task.	12
Help user recognize, diagnose and recover from errors	The "Query search not found" error message displayed is helpful in rectifying any errors	8
Help and documentation	This system can be operated and navigated with little / no instruction and help documentation.	2

Table 2. Heuristic Evaluations Result Summary

Responses towards the overall impression of the QGV show that the positive features of the QGV focused on the recommended query lists which gave users more ideas on what query terms to use, and the graph visualization scheme which specifies the relationship between queries with varying weights. Participants also liked the options to control the visualization area. For example, one participant said that the QGV was “really useful in speeding up the searching process and to better understand the relation between queries, since the weights between queries are clearly shown. It also made searching process more fun because of the graph and the zooming features.

The system is also easy to use and the functions are self explanatory, which can be followed with minimum instruction”. The negative comments go to the lack of detailed documentation which was needed to support the successful use of the QGV by some participants. This was due to the fact that some technical terms were used in the system, such as “weight”, “locality zooming”. Further, three participants complained that the QGV was time consuming to load and execute because it needed to be downloaded into client’s browser before it could be run. Despite the fact that there are some drawbacks in the current version of QGV, all 12 participants expressed strong interest in it, and indicated that they would use the QGV if it became available in the future.

5 Conclusions and Future Work

In this paper, we introduced a search assistant, the QGV, which utilizes the hybrid query similarity measure to generate query clusters for each submitted query. Our work can contribute to research in collaborative querying systems that mine query logs to harness the domain knowledge and search experiences of other information seekers found in them. Firstly we propose a hybrid query clustering approach which differs from [7, 9, 11] since all them do not use the query content itself. Secondly, we employ a graph-based approach to visualize the recommended queries which differs from [1, 9] since all them only adopt text or HTML to display the recommended queries.

Our evaluation examined the effectiveness of the QGV from the perspectives of usefulness and usability. The time to complete tasks and the subjective preferences of the participants were used as metrics in our study. Results show that the system offers a viable option for online information seeking. The QGV appears to help users complete information seeking tasks faster and our participants expressed a preference for using the system. However the QGV might not be useful for all types of information seeking tasks since the tasks in our study are open-ended and required some amount of exploration. Whether users can benefit from the QGV for well-defined search tasks remains unknown.

In addition to the initial development in this work, alternative approaches to detecting related queries will also be attempted. In the existing algorithm, if a new query does not exist in the query repository, the QGV will display statistical information regarding with each term in the query. Possible solutions to improve this basic approach include using other query clustering algorithms, such as K-means, to detect related queries. Further, our evaluation also highlighted several areas for further improvement. For example, because the graph-based mode of interaction is unfamiliar to many users, online help will need to be incorporated into the QGV. Finally, due to the small sample size of this initial evaluation, our findings cannot be generalized. Instead, a comprehensive evaluation will be conducted to further assess the performance and effectiveness of the QGV involving more users and a greater variety of task types.

Acknowledgements

This project is partially supported by NTU with the research grant number: RCC2/2003/SCI. Further we would like to express our thanks to the NTU Library and the Centre for Information Technology Services for providing access to the queries.

References

- [1] Anick, P.G. & Tipirneni, S. (1999) The paraphrase search assistant: Terminological feedback for iterative information seeking. *Proceedings of SIGIR 99*, 153-161.
- [2] Anick, P. G. (2003) Using terminological feedback for web search refinement: A log-based study. *In proceedings of SIGIR '03*, 88-95.
- [3] Bruza, P.D., Dennis, S. (1997) Query reformulation on the Internet: Empirical data and the Hyperindex search engine. *Proceedings of the RIAO 97 Conference*, 488-499.
- [4] Churchill, E.F., Sullivan, J.W. & Snowdon, D. (1999) Collaborative and co-operative information seeking, *CSCW'98 Workshop Report* 20(1), 56-59.
- [5] Crouch, C.J., Crouch, D.B. & Karedy, K.R. (1990) The automatic generation of extended queries, *Proceedings of the 13th Annual International ACM SIGIR Conference*, 269-283.
- [6] Ester, M., Kriegel, H., Sander, J., Xu, X., (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of second International Conference on Knowledge Discovery and Data Mining*, 226-231.
- [7] Fitzpatrick, L. & Dent, M. (1997). Automatic feedback using past queries: Social searching? *Proceedings of SIGIR '97*, 306-313.
- [8] Fu, L. Goh, D. & Foo, S. (2003). Collaborative querying through a hybrid query clustering approach. *Proceedings of Sixth International Conference of Asian Digital Libraries*, 111-122.
- [9] Glance, N. S. (2001). Community search assistant. *Proceedings of Sixth ACM International Conference on Intelligent User Interfaces*, 91-96.
- [10] Nielsen, J. 1992. Finding usability problems through heuristic evaluation. *In Proceedings of the ACM CHI'92 Conference*, 373-380.
- [11] Raghavan, V. V., & Sever, H. (1995). On the reuse of past optimal queries. *Proceedings of the Eighteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*, 344-350.
- [12] Salton, G. & McGill, M.J. (1983). *Introduction to Modern Information retrieval*. McGraw-Hill New York, NY.
- [13] Silverstein, C., Henzinger, M., Marais, H., & Moricz, M. (1998) Analysis of a very large Altavista query log. *DEC SRC Technical Note 1998-14*.
- [14] Wen, J.R., Nie, J.Y., & Zhang, H.J. (2002) Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1), 59-81.