Collaborative querying for enhanced information retrieval

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

2004

Fu, L., Goh, D., Foo, S., & Supangat, Y. (2004). Collaborative Querying for Enhanced Information Retrieval. Proceedings of the 8th European Conference on Digital Libraries ECDL 2004, (September 12-17, Bath, UK), Lecture Notes in Computer Science 3232, 378-388.

<https://hdl.handle.net/10356/91556>

<https://doi.org/10.1007/b100389>

Fu, L., Goh, D.H., Foo, S., and Supangat, Y. (2004). Collaborative querying for enhanced information retrieval. *Proceedings of the 8th European Conference on Digital Libraries ECDL 2004*, (September 12-17, Bath, UK), *Lecture Notes in Computer Science* 3232, 378-388.

Collaborative Querying for Enhanced Information Retrieval

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. Communication and collaboration with other people is a major theme in the information seeking process. Collaborative querying addresses this issue by sharing other users' search experiences to help users formulate appropriate queries to a search engine. This paper describes a collaborative querying system that helps users with query formulation by finding previously submitted similar queries through mining web logs. The system operates by clustering and recommending related queries to users using a hybrid query similarity identification approach. The system employs a graph-based approach to visualize the query recommendations.

1 Introduction

Information seeking is a broad term encompassing the ways individuals articulate their information needs, seek, evaluate, select and use information. In the course of a search, the individual may interact with people, manual information systems (such as libraries) or with digital libraries. A major theme in the various information seeking models is that interaction and collaboration with other people is an important part in the process of information seeking and use (e.g. [13] [14]).

Given this idea, collaborative querying aims to assist users in formulating queries to meet their information needs by harnessing other users' expert knowledge or search experience [6] [17]. A common approach in collaborative querying is known as query clustering, which is to group similar queries automatically without using predetermined class descriptions. Such queries are typically stored in user logs, which are then extracted and clustered to obtain recommended queries to users. 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 propose 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. Further we describe a prototype collaborative querying system 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 collaborative querying system. A scenario is given to highlight the usefulness of this system. Finally, we discuss the implications of our findings for collaborative querying systems and outline areas for further improvement.

2 Related Work

There are several useful strands of literature that bear some relevance to this work. This section 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.

2.1 Interactive query reformulation systems

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. Several techniques have been used to incorporate aspects of interactive query reformulation systems into the information retrieval process.

One approach to obtain the recommended queries is to use terms extracted from the search result documents. Examples include HiB [5], Paraphrase [1] and Altavista Prisma [2], which parse the list of result documents and use the most frequently occurring terms as recommendations. Some popular commercial search engines, such as Altavista [3], Askjeeves [4], Eurekster [9], etc, incorporate term recommendation functions in the hope that it can help users reformulate an accurate query and then locate relevant content.

Another approach is collaborative querying. Related queries (the query clusters) may be calculated based on the similarities of the queries in the query logs [12] which provide a wealth of information about past search experiences. The system can then either recommend the similar queries to users [12] or use them as expansion term candidates to the original query to augment the quality of the search results [7]. Here, calculating the similarity between different queries and clustering them automatically are crucial steps. This will be discussed in the next section.

2.2 Query Clustering Approaches

Traditional information retrieval research suggests an approach to query clustering by comparing query term vectors (content-based approach). Various similarity functions are available including cosine-similarity, Jaccard-similarity, and Dice-similarity [16]. Using these functions have provided good results in document clustering due to the large number of terms contained in documents. However, the content-based method might not be appropriate for query clustering since most queries submitted to search engines are quite short [21]. A recent study on a billion-entry set of queries to AltaVista has shown that more than 85% queries contain less than three terms and the average length of queries is 2.35 [18]. 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.

Raghavan and Sever [15] determine similarity between queries by calculating the overlap in documents returned by the queries. This is done by converting the query result documents into term frequency vectors. The similarity between two queries is then decided by comparing these vectors. Fitzpatrick and Dent [10] further developed this method by weighting the query results according to their position in the result list. They argue that the beginning of a result list is more likely to include a relevant document to the original query. Using the corresponding query results is useful in boosting the performance of query clustering in terms of precision and recall [10, 15]. However this method is time consuming to execute [15]. Glance [12] thus uses the overlap of result URLs as the similarity measure instead of the document content. Queries are posted to a reference search engine and the similarity between two queries is measured using the number of common URLs in the top 50 results list returned from the reference search engine.

3 A Collaborative Querying System

We have designed a collaborative querying system based on the query clusters generated using the hybrid query similarity measure. 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 collaborative querying system. After capturing a new query, the system will search for matching documents, which is similar in function to traditional information retrieval systems. However, beyond the search results, the system will identify related queries and use them as recommended queries to users. The recommended queries are displayed together with the search results, similar to [1, 2, 5]. Users may further explore the recommended queries by visualizing the query clusters which contains the initial query and recommended

queries. Our query graph visualizer is designed to be an independent agent and can be incorporated into different information retrieval systems. Put differently, our collaborative querying system can provide additional information that a user is originally unaware of so that the user can use it to formulate a better query to express his/her information needs.

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 query recommendation phase that includes related query detection and query graph visualization. The third process the maintenance of the query repository.

3.2 Query Repository Construction

In this phase, we need to cluster related queries and save the query clusters into the query repository. As discussed, our approach to query clustering uses a hybrid method based on the analysis of query terms and query results. Here, two queries are similar when (1) they contain one or more terms in common (content-based approach); or (2) they have results that contain one or more items in common (result-based approach). The remainder of this section provides definitions of different query similarity

measures used in our experiments. Our method of constructing query clusters based on different query similarity measures is also presented.

The content-based approach clusters queries by calculating the overlap of identical terms between queries. Taking the term weights into consideration, we can use any of the standard similarity measures [16]. Here we only present cosine similarity measure since it is most frequently used in information retrieval.

$$Sim_cosine(Q_i, Q_j) = \frac{\sum_{i=1}^k cw_{iQ_i} \times cw_{iQ_j}}{\sqrt{\sum_{i=1}^k cw_{iQ_i}^2} * \sqrt{\sum_{i=1}^k cw_{iQ_j}^2}} \quad (1)$$

where cw_{iQ_i} refers to the weight of i^{th} common term between Q_i and Q_j in query Q_i and cw_{iQ_i} is calculated by TFIDF.

The results-based approach uses the overlap of result URLs as the similarity measure instead of the query content as shown in formula (2). The results returned by search engines usually contain a variety of information such as the title, abstract, topic, etc. This information can be used to compare the similarity between queries. In our work, taking the cost of processing query results into consideration, we consider the query results' unique identifiers (e.g. URLs) in determining the similarity between queries.

$$Sim_result(Q_i, Q_j) = \frac{|R_{ij}|}{Max(|U(Q_i)|, |U(Q_j)|)} \quad (2)$$

where the $|U(Q_i)|$ is the number of result URLs for Q_i , and $|R_{ij}|$ is the number of common result URLs between Q_i and Q_j .

For the content-based approach, a single query term can represent different information needs. For the result URLs-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. Hence, the hybrid approach is expressed as:

$$Sim_hybrid(Q_i, Q_j) = \alpha * Sim_result(Q_i, Q_j) + \beta * Sim_cosine(Q_i, Q_j) \quad (3)$$

where α and β are parameters assigned to each similarity measure, with $\alpha + \beta = 1$.

Two queries are in one cluster whenever their similarity is above a certain threshold. We construct a query cluster G for each query in the query set using the definition in (4).

$$G(Q_i) = \{Q_j : Sim(Q_i, Q_j) \geq threshold\} \quad (4)$$

where $1 < j < n$; n is the total number of query.

Note that there are alternative clustering algorithms besides the one used in our experiments [8]. Compared with these approaches, our method is relatively less time consuming.

In order to test the usefulness of the hybrid query clustering approach, we collected 20000 queries from the digital library at Nanyang Technological University (Singapore). After preprocessing the original queries, including stop word removal, misspelled term checking, etc, there were 16000 queries for our experiments.

We generated different sets of query clusters based on different approaches. Computation for the similarity between two queries based on query content (sim_cosine) was straightforward using function (1). For sim_result, we posted each

query to a reference search engine (Google) and retrieved the corresponding result URLs, similar to [11]. Since search engines rank highly relevant results higher, we only considered the top 10 result URLs returned to each query. The result URLs were then be used to compute the similarity between queries according to function (2). For the hybrid approach (sim_hybrid), the issue was to determine the values for the parameters α and β . We used pairs of α and β with the following values respectively: (0.25, 0.75), (0.5, 0.5) and (0.75, 0.25). Due to space constraints, we only report results for $\alpha=0.25$ and $\alpha=0.75$ since this pair of values generates the best quality query clusters.

Recall that the threshold is the minimum value, obtained from a given similarity measure, that determines whether two queries should be clustered into to the same group. Here, thresholds were set to 0.25, 0.5, 0.7 and 0.9.

In our experiments, the quality of query clusters is measured using the F-measure [21]. The F-measure used here examines the overall quality of query clusters by combining precision and recall, with the value varying from 0 to 1. The larger the F-measure value, the better the quality of query cluster. Figure 2 shows the F-measure values of the three approaches.

Along with the change of threshold from 0.25 to 0.9, the F-measure value of sim_hybrid increases from 56% to 77%, sim_cosine increases from 49% to 52% and sim_result decreases from 21% to 20%. We see that sim_hybrid generates the best results comparing with sim_cosine and sim_result. This confirms our hypothesis that a combination of both query terms and result URLs provide a better quality of query clusters than using each separately. More experimental results can be found in [11]

Fig. 2. F-measure for different approaches

3.3 Query Recommendation

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

3.3.1 Detecting Related Queries

Given a query submitted by a user, we first search the query repository for related queries. These queries are then recommended to the user. Here a recursive algorithm was implemented to search the query clusters in the query repository and the initial query will be regarded as 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 3. 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 $G(Q_i)$, therefore the system will compute the query cluster $G(Q_1)$, which forms the second level as shown in Figure 3. This process is iterative and 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 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 .

Fig. 3. Detecting related queries

3.3.2 Query Cluster Visualization

Our system displays query clusters in a graph (Figure 4). 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” shows 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 levels of the 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. digital library at Nanyang Technological University). Recall that the query graph visualizer is running as an independent agent and can be incorporated into various search engines. Secondly, users may use this query to carry out another round of searches across the query repository and detect queries related to the selected one. Further, users can expand and collapse each query node on the graph. Note the number beside each node that denotes how many child nodes that have not been expanded yet.

The query graph visualizer was implemented using “Touchgraph” which is an open source component to visualize information in graph formats [20].

Fig. 4. Query graph visualizer

3.4 Updating Query Repository

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. Figure 5 shows the steps of updating the query repository. Most of the steps are similar to the query repository construction process except the first one – capturing the new queries. This means the newly submitted queries will be captured and compared with existing queries and incorporated into the query database only if they are unique. For the rest of the steps, refer to Section 3.2.

Fig. 5. Updating Query Repository

4 A Scenario of Use

The following scenario illustrates one of the potential users of the system and highlights the operation of the system.

Suppose that a user is interested in the field of data mining and he is a novice in this area. When he uses the collaborative querying system, the user first submits a query “data mining” to search for information. A moment later, a list of queries related to “data mining” is displayed as the query recommendations in addition to the search results. After looking through the result list and the recommended queries, he wants to generate a query graph using “data mining” as the root node. Thus the user triggers the query cluster visualizer. A query graph will appear on the visualization

area (see Figure 4 for an example). While browsing the graph, he is interested in the node “knowledge discovery”. It is a new phrase to him but seems related to his search topic. Wanting to peruse the queries related to “knowledge discovery”, he zooms in the visualization area by dragging the bar next to the option box from left to right (see Figure 6-(a) and 6-(b)). He may also rotate the visualization area to facilitate his browsing (see Figure 6-(b) and 6-(d)). By adjusting the locality level, the user expands or collapses the nodes that contain child nodes in order to obtain an overview about the whole structure of all the queries related to the root node (see Figure 6-(a) and 6-(c)).

Fig. 6. Query cluster visualization

Now the user notices that there is a number “3” near the node “data warehousing, data mining and OLAP” (see Figure 6-(d)). The number here indicates that this node has two child nodes which have not been expanded. He right clicks on the node and chooses ‘expand this node’ on the popup menu. Note this action will only affect the selected node while the locality zooming option discussed previously take effect

across the whole visualization area. After examining the graph carefully, the user is prepared to carry out another round of information retrieval by using the node “knowledge discovery”. He thus right clicks on the node and chooses “display result in a separate browser”. The query “knowledge discovery” will be posted to the search engine automatically and the results will be displayed in a separate browser. He may repeat this process until he finds the desired information.

5 Conclusions and Future Work

In this paper, we first compared different query similarity measures. Our experiments show that by using a hybrid content-based and results-based approach, considering both query terms and query result URLs, better query clusters can be generated than using either of them alone. We then introduced a collaborative querying system which utilizes the hybrid query similarity measure to generate query clusters for each query. We described the design and implementation of a collaborative querying system based on the query clusters. 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 [10, 12, 15] 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, 3, 4, 12] since all them only adopt text or HTML to display the recommended queries.

In addition to the initial experiments performed in this research, alternative approaches to identifying the similarity between queries will also be attempted. Adaptive elements will be introduced to reflect the growing and changing nature of the collection of documents to ensure the quality of query clusters when using the results-based approach. In addition, word relationships like hypernyms can be used to replace query terms before computing the similarity between queries. Finally, a user evaluation to test the usefulness and usability of the collaborative querying system will be conducted.

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 at NTU 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] Altavista home page. Retrieved on March, 4, 2004 from <http://www.altavista.com>
- [4] Askjeeves home page. Retrieved on March 9, 2004 from <http://www.ask.com>
- [5] 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.
- [6] Churchill, E.F., Sullivan, J.W. & Snowden, D. (1999) Collaborative and co-operative information seeking, *CSCW'98 Workshop Report* 20(1), 56-59.
- [7] 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.
- [8] 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.
- [9] Eurekster home page. Retrieved on March 15, 2004 from <http://www.eurekster.com>
- [10] Fitzpatrick, L. & Dent, M. (1997). Automatic feedback using past queries: Social searching? *Proceedings of SIGIR '97*, 306-313.
- [11] 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.
- [12] Glance, N. S. (2001). Community search assistant. *Proceedings of Sixth ACM International Conference on Intelligent User Interfaces*, 91-96.
- [13] Lokman, I. M., & Stephanie, W. H. (2001) Information-seeking behavior and use of social science faculty studying stateless nations: A case study. *Journal of library and Information Science Research*, 23(1), 5-25.
- [14] Marchionini, G. N. (1995). *Information seeking in electronic environments*. Cambridge, England: Cambridge University Press.
- [15] 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.
- [16] Salton, G. & McGill, M.J. (1983). *Introduction to Modern Information retrieval*. McGraw-Hill New York, NY.
- [17] Setten, M.V & Hadidiy, F.M. Collaborative Search and Retrieval: Finding Information Together. Available at: https://doc.telin.nl/dscgi/ds.py/Get/File-8269/GigaCE-Collaborative_Search_and_Retrieval_Finding_Information_Together.pdf
- [18] Silverstein, C., Henzinger, M., Marais, H., & Moricz, M. (1998) Analysis of a very large Altavista query log. *DEC SRC Technical Note 1998-14*.
- [19] Taylor, R. (1968). Question-negotiation and information seeking in libraries. *College and Research Libraries*, 29(3), 178-194.
- [20] Touchgraph website. Retrieved on January, 1, 2004 from <http://touchgraph.sourceforge.net>
- [21] Wen, J.R., Nie, J.Y., & Zhang, H.J. (2002) Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1), 59-81.