Collaborative querying using the Query Graph Visualizer

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

2005

Goh, D., Fu, L., & Foo, S. (2005). Collaborative querying using the query graph visualizer. *Online Information Review*, 29(3), 266-282.

<https://hdl.handle.net/10356/91658>

<https://doi.org/10.1108/14684520510607588>

Online Information Review @ copyright 2005 Emerald. The journal's website is located at
<http://www.emeraldinsight.com/Insight/viewContentItem.do?contentType=Article&contentId=1509086>.

Downloaded on 26 Apr 2023 21:17:06 SGT

Goh, D.H., Fu, L., and Foo, S. (2005). Collaborative querying using the query graph visualizer. *Online Information Review*, 29(3), 266-282.

Collaborative Querying using the Query Graph Visualizer

Dion Hoe-Lian Goh

Division of Information Studies
School of Communication & Information
Nanyang Technological University
Singapore 637718
SINGAPORE
E-mail: ashlgoh@ntu.edu.sg

Dion Goh is an Assistant Professor at the School of Communication and Information, Nanyang Technological University, Singapore

Lin Fu

Division of Information Studies
School of Communication & Information
Nanyang Technological University
Singapore 637718
SINGAPORE
E-mail: p148934363@ntu.edu.sg

Fu Lin is a PhD candidate in the Division of Information Studies, School of Communication and Information, Nanyang Technological University, Singapore

Schubert Shou-Boon Foo

Division of Information Studies
School of Communication & Information
Nanyang Technological University
Singapore 637718
SINGAPORE
E-mail: assfoo@ntu.edu.sg

Schubert Foo is a Professor and Vice-Dean of the School of Communication and Information, Nanyang Technological University, Singapore

Collaborative Querying using the Query Graph Visualizer

Keywords

Collaborative querying, information retrieval, query clustering, query networks

Abstract

Information overload has led to a situation where users are swamped with too much information, resulting in difficulty sifting through material in search of relevant content. We address this issue from the perspective of collaborative querying, an approach that helps users formulate queries by harnessing the collective knowledge of other searchers. We describe the design and implementation of the Query Graph Visualizer (QGV), a collaborative querying system which harvests and clusters previously issued queries to form query networks that represent related information needs. The queries in the network are explored in the QGV, helping users locate other queries that might meet their current information needs. A preliminary evaluation of the QGV is also described and results suggest the usefulness and usability of the system.

Introduction

Most information retrieval (IR) systems are developed using the “best match” principle (Belkin, Oddy and Brooks, 1982) which assumes that users can specify their information needs in a query. Using this principle, the IR system retrieves documents matching closely to the query and regards these documents as relevant to the user. Information seeking researchers have longed argued that problems exist when using this approach. Examples

include unfamiliarity among users of an IR system's operations, the vocabulary mismatch between indexers and users, and the contextual nature of human judgment.

Much research has been done to address these problems inherent in information search on the Web. One such area is in the development of innovative algorithms used in indexing and retrieval. Examples include Kleinberg's (1998) HITS (Hypertext-induced Topic Selection) and Google's PageRank algorithms that employ link analysis (Page and Brin, 1998). Another approach to improving the performance of search engines is by helping users refine their queries through automatic query expansion, in which the system attempts to add additional terms to the original query (Smeaton and van Rijsbergen, 1983). Other approaches focus on user interface techniques such as search results visualization (e.g. Roussinov, Tolle, Ramsey, McQuaid and Chen, 1999) and graphical query reformulation (e.g. Kovács, Micsik and Pataki, 1999). Yet another approach involves the organization of information, in which human experts judge the relevance of Web sites (e.g. Yahoo and the Open Directory Project).

Research in information seeking behavior however suggests an alternative approach in helping users meet their information needs. Studies have found that interaction and collaboration with other people is an important component in the process of information seeking and use. For example Taylor's (1968) model highlights the importance of the interaction between the inquirer and the librarian, while Ellis (1993) argues that communication with colleagues is a key component in the initial search for information. It is thus not uncommon that in searching for information, we tap on our social networks – friends, colleagues, librarians, etc., to help locate what we need.

We use the term collaborative querying (Fu, Goh, Foo and Na, 2003) to refer to a family of techniques that assist users in formulating queries to meet their information needs by harnessing other users' expert knowledge or search experience. On the Web, elements of collaborative querying can be seen in the FAQ pages found on many Web sites. In addition, with the increasing popularity of Google, people have also begun to retrieve information via sharing of queries (e.g. "enter this query on Google and it is in the first document on the results listings"). Further, sites such as MetaSpy (<http://www.metaspay.com>) provide continuously updated lists of queries submitted to the MetaCrawler search engine for visitors to view. Taking query-watching a step further, queries are also the subject of analysis and commentary. The Lycos 50 (<http://50.lycos.com>) and the Google Zeitgeist (<http://www.google.com.sg/press/zeitgeist.html>) are some examples of discussions of the most popular queries submitted to their respective search engines.

Queries, being expressions of information needs, thus have the potential to provide a wealth of information that could be used to guide other searchers with similar information needs, helping them with query reformulation. If large quantities of queries are collected, they may be mined for emerging patterns and relationships that could be used to define communities of interest (e.g. Lycos 50), and even help users explore not only their current information needs but related ones as well. One can imagine a network of queries, amassed from the collective expertise of numerous searchers representing what people think as helping them meet their information needs. Query formulation and reformulation will then be a matter of exploring the query network, executing relevant queries, and retrieving the resulting documents.

In this paper, we present the Query Graph Visualizer (QGV), a collaborative querying system employing a graph-based visualization interface for exploring query networks. The QGV

operates by mining transaction logs of previously submitted queries, clustering them, and presenting these clusters to users for exploration. The remainder of this paper is organized as follows. In the next section, we review the literature related to this work. We then discuss our approach to collaborative querying, describing the design, implementation and use of the QGV. Finally, we highlight findings from an initial evaluation of the system, discuss the significance of our work, and outline areas for future research.

Related Work

Interactive Query Reformulation Systems

With the increasing popularity of search engines and the proliferation of documents on the Web, users commonly encounter difficulty in formulating queries that express their information needs. Interactive query reformulation systems aim to address this problem by identifying information needs through submitted queries and giving users opportunities to rephrase their queries by suggesting alternative queries.

One approach to query reformulation is to suggest new query terms extracted from the search results documents (Smeaton & van Rijsbergen, 1983). Using this method, a query is run using conventional information retrieval techniques. Related terms are then extracted from, for example, the top 10 documents that are returned in response to the original query while additional terms are selected using a variety of heuristics. The related terms are then added to the original query, and the expanded query is run again to generate a fresh set of documents, which are returned to the user. Examples include HiB (Bruza and Dennis, 1997) and Altavista Prisma (Anick, 2003), which parse the list of result documents and use the most frequently occurring terms as recommendations. Kraft and Zien (2004) also proposed a technique to automatically expand a user's query by mining the anchor text of hyperlinks in HTML pages.

The query refinement process involves examining the original query, finding all anchor texts that are similar and presenting these to the user as query refinements.

Another approach to query reformulation is collaborative querying. This method typically depends on the search history found in transaction logs maintained by the IR system, allowing a user to utilize the previous searches of other users to meet their information needs. This search history can be visualized in a manner that can easily be interpreted by others. Besides viewing, a user can even replay parts of the search process during his/her own search using different parameters (Setten and Hadidy, 2000). An example of a collaborative querying system is the Community Search Assistant (Glance, 2001). The Community Search Assistant is a software agent that helps users in producing a suitable query string for a particular search session by showing relevant queries posed by other users. All queries submitted by the community of users are stored in a network format in which links are made between queries found to be related. In addition, Billerbeck, Scholer, Williams and Zobel (2003) proposed a method for expansion terms based on selecting terms from previously submitted queries that are associated with documents in the collection. Here, past queries are stored as document surrogates for documents statistically similar to the query. These queries are then used as term recommendations for similar newly submitted queries. Their experiments show that performance is 26% to 29% more effective than with no term expansion.

Query Similarity Computation Techniques

A crucial component in the identification of related queries is the notion of similarity between queries. Once computed, related queries can then be clustered and then used for query

recommendation or term expansion. A variety of similarity computation techniques are available as suggested by the literature.

Term-based. Traditional information retrieval research (Salton and McGill, 1983) suggests an approach that compares query term vectors using a similarity function such as the cosine, Jaccard, and Dice measures. Wen, Nie and Zhang (2002) adopted this approach and used a simple measure that counted the number of overlapping terms. However, term-based methods might not be appropriate for query clustering especially when used alone, since most queries submitted to search engines are quite short, typically between 2 to 3 terms (e.g. Jansen, Spink and Saracevic, 2000). Put differently, query terms alone are not able to adequately convey much information nor help to detect the semantics behind queries since the same term might represent different meanings.

Results-based. Here, similarity is determined by comparing the attributes of documents returned by the queries. For example, Raghavan and Sever (1995) converted documents into term frequency vectors and compared them to determine similarity between queries. Glance (2001) employed a less computationally expensive approach and used the overlap of the top 50 search results URLs retrieved from a reference search engine as the similarity measure instead of the document content. However, a pure results-based approach ignores the original query terms that could provide useful clues for determining similarity between queries.

Feedback-based. This technique uses the information contained in “clickthrough data” for clustering queries (Beeferman and Berger, 2000). Previously submitted queries in transaction logs are analyzed to determine the documents users selected for a given query and these are treated as a measure of similarity. The underlying principle is that if two documents are

judged relevant to the same query, then there is reason to believe that these documents discuss the same topic, and therefore can be included in the same query cluster (Wen, Nie and Zhang, 2001). While promising, a major limitation is that if a user indiscriminately selects on a large number of irrelevant documents, poor similarity computation results will be obtained.

Community-based. This may be considered a refinement of the above techniques. Rather than considering the universe of all possible query terms, search results documents and clickthrough data, community-based approaches only consider a subset gathered from relevant communities of interest. The idea is that similarity is better judged by people with the same interests rather than by the entire user population who have varied interests. Thus, if a person is interested in data mining and knowledge discovery in databases, the queries and document clicks of people with the same interest would be more relevant than those interested in gold mining. A major issue is the need to identify such communities. The search engine Eureka (http://www.eureka.com) for example requires users to explicitly create such communities (search groups) or join existing ones.

The Query Graph Visualizer

The QGV is a collaborative querying system that harvests previously submitted queries, clusters them and presents these clusters as query networks to users for visualization, exploration and query reformulation. As its name suggests, the QGV displays query clusters in a graph (see Figure 1). Each graph node represents a single query and edges between nodes show the relationship between two queries, with the value on the edge indicating the strength of the relationship. For example, 0.3 on the edge between the nodes “data mining” and “knowledge discovery” indicates that the similarity weight between these two nodes is 0.3, given some similarity computation that will be discussed later.

Figure 1. The Query Graph Visualizer.

In the QGV, the user's submitted query is known as the root node and is rendered in the center of the visualization area. Nodes that are directly connected to the root node represent a cluster of related queries. Each query in the cluster may be related to other queries, and thus connected to a different query cluster. This relationship between queries and clusters thus forms a network, in which a query is directly or indirectly related to other queries. To better differentiate between nodes, different colors are used for different levels of nodes from the root node. Specifically, lighter node colors are used for levels further away from the root. Different colors are also used for edges to denote the strength of the relationship between for nodes. Here, the stronger the relationship value of the edge, the darker the color. Figure 2

shows a network for the submitted query “data mining”. This query is directly related to queries such as “predictive data mining” and “data warehousing, data mining and OLAP”. The latter in turn is related to “jiawei han”, a noted researcher in the field, indicating that “data mining” is also related to “jiawei han”. This approach therefore allows users to explore new query formulations that are diverse, sometimes unexpected, and potentially useful.

Figure 2. A query network for “data mining”.

The QGV provides a tool bar to manipulate the visualization area with features such as zooming, rotating and locality zooming (see Figure 1). The zooming function allows users to shrink or enlarge the visualization area. The rotating function allows users to view the visualization area from different directions. Locality zooming allows the number of levels of related queries to be displayed, starting from the root node’s cluster. Thus for example, a level 1 locality zoom displays the root node and its cluster of related queries, while a level 2

locality zoom will show the root node, its cluster of related queries, and the related queries' clusters. Other options available include setting the number of query nodes to be displayed at each level, and setting the maximum depth level and the minimum weight limit for displaying relationships between nodes.

By right clicking on an individual node in the visualization area, a popup menu offering a variety of options will appear (see Figure 3). Firstly, users can post a selected query node to a search engine to retrieve its associated documents. Users may also make the selected query the root node, causing the QGV to construct a new query network around it. Further, users can expand and collapse each query node on the graph. Note the number beside each node in Figure 3 which denotes how many child nodes that have not been expanded yet. These numbers appear when there are more levels of related nodes than the current locality zoom level is set to display.

Figure 3. Enlarged view of node options.

A Scenario of Use

Suppose that a user is interested in the field of data mining. He is new to this area and would like to explore and learn related topics. Upon visiting his favorite search engine, the user first submits a query “data mining” to begin his search. A moment later, a list of search results appears. At the same time, the user notices a new feature on the search engine which invites him to launch the QGV to explore related searches. Accepting the invitation, the QGV is executed and a query network soon appears, showing “data mining” as the root node with related queries surrounding it (see Figure 4(a)). By adjusting the locality level, the user can also expand or collapse the nodes that contain child nodes in order to obtain a high-level

overview or in-depth examination of the entire structure of queries related to the root node “data mining” (see Figure 4(b)).

Figure 4. Using the QGV to explore the query network.

Suppose while browsing the network, the user becomes interested in the node “knowledge discovery” as it is a new term to him but appears related to his search. Wanting to examine the queries related to “knowledge discovery” further, he zooms and pans the visualization area, expanding child nodes as well, until this node becomes the focus (see Figure 5(a)). Finally, deciding that this query looks promising, he makes “knowledge discovery” the root node by right-clicking on it and selecting the appropriate option, causing the QGV to

construct a query network around it (see Figure 5(b)). At the same time, the user also submits this query to his search engine to peruse its associated documents. This process may be repeated until the desired information is found.

Figure 5. Using the QGV to explore the query network (continued).

QGV Architecture

Figure 6 presents a high-level overview of the QGV's architecture. The QGV is not meant to be a replacement for IR systems, but is designed to be a complementary, value-added module that can be incorporated into various IR systems. As shown in the figure, there are two distinct sections – the IR component and the QGV component, both working in tandem. A

user submits a query via an IR search interface. Behind the scenes, the query is executed by the IR engine to retrieve a set of relevant documents which are displayed to the user. At the same time, the query is routed to the QGV's query network retrieval engine where it is matched with an existing cluster. A recursive process is executed to retrieve indirectly related query clusters as well. The entire network is then returned to the QGV for display. As the user interacts with the QGV, any query found in the network can be executed, repeating the entire process.

Figure 6. The architecture of the QGV.

The QGV is a Java-based client/server system. The user interface component is implemented in Swing and Touchgraph (<http://toughgraph.sourceforge.net>), an open source graph framework for visualizing information in graphs. The server component consists of Java servlets that receive query requests from the client and sending query networks to the client. Data sent between client and server is via XML over HTTP. For this purpose, Sun's JAXP (Java API for XML Processing) and the Apache Xerces Parser are used for processing XML

data. Query clustering is performed using Java programs written for this research. In addition, the system requires an ODBC-compliant database to manage the query repository.

Computing Query Similarity

In order to construct the query repository, access to the IR system's transaction logs of previously submitted queries is required. We extract these queries, compute their similarity values, cluster them and save these clusters into the query repository. As discussed previously, there are four major techniques that may be used to compute similarity between queries: term-based, results-based, feedback-based and community-based. While each of the individual techniques offers distinct advantages, we adopt instead a combination of techniques in calculating similarity, and call this the hybrid approach. Here, we hypothesize that the strengths of one approach can compensate for the weaknesses inherent in another (Fu, Goh, Foo and Na, 2003). In our work, we use a linear combination of the term-based and results-based approaches. That is, two queries are similar when (1) they contain one or more query terms in common (term-based approach); or (2) they have results that contain one or more documents in common (result-based approach). Specifically, the hybrid approach to computing similarity (sim_hybrid) between two queries Q_i and Q_j is defined as:

$$sim_hybrid(Q_i, Q_j) = \alpha * sim_result(Q_i, Q_j) + \beta * sim_term(Q_i, Q_j)$$

where

- $sim_result(Q_i, Q_j)$ is the similarity between Q_i, Q_j using the results-based approach
- $sim_term(Q_i, Q_j)$ is the similarity between Q_i, Q_j using the term-based approach
- α and β are parameters assigned to each similarity measure, with $\alpha + \beta = 1$.

In the results-based portion of our hybrid similarity measure, we use the number of overlapping search results document identifiers (URLs) to determine the degree of similarity, similar to Glance (2001):

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

where

- $|U(Q_i)|$ and $|U(Q_j)|$ are the number of result URLs for Q_i and Q_j respectively
- $|R_{ij}|$ is the number of common result URLs between Q_i and Q_j

In the term-based portion, we treat each query as a term vector and employ the cosine measure commonly used in information retrieval to calculate similarity:

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

where

- cw_{iQ_i} and cw_{iQ_j} refer to the weights of i^{th} common term between Q_i and Q_j respectively
- the weights are calculated by the standard TFIDF formula

Finally, the output of each approach contributes a certain percentage of the total similarity between two queries. The actual values of α and β used is discussed in the next section.

We wanted a similarity computation technique that could be applicable to a wide variety of IR systems. Thus, our criteria for the choices made for constructing query networks was simplicity and expected availability of data. As such, we did not use the feedback-based approach because not all IR systems provide access to clickthrough data. The community-based approach was similarly not considered because additional effort would be needed either by users to join communities or by the system to automatically identify such communities.

Query Repository Construction and Maintenance

Given the similarity values between queries, we next create clusters using the following rule: two queries are in one cluster whenever their similarity is above a certain threshold. We thus construct a query cluster G for each query in the repository using the following definition:

$$G(Q_i) = \{Q_j : \text{sim_hybrid}(Q_i, Q_j) \geq \text{threshold}\}$$

where

- $G(Q_i)$ refers to the cluster of query Q_i
- Q_j refers to the members of the cluster
- n is the total number of queries, $1 < j < n$

Our approach to query clustering and hence query repository construction requires three parameters to be set:

- α, β – parameters assigned to each similarity measure
- threshold – the similarity threshold for cluster membership

Using data from 20000 queries obtained from the Nanyang Technological University (Singapore) digital library, our experiments reveal that when $\alpha = 0.25$, $\beta = 0.75$ and threshold = 0.9, the best quality clusters are obtained. The selection criteria include F-measure, coverage and average cluster size. Put succinctly, for any given query submitted by a user, he/she will have a high probability of obtaining a query network that was diverse and yet relevant to the current information need expressed by the original query. More information about these experiments can be found in Fu, Goh, Foo and Na, (2003) and Fu, Goh and Foo (2004).

Since queries within the same cluster are related given their similarity values, a network can be formed, with nodes representing the queries and edges representing their degree of

relatedness. Further, because our clustering technique produces non-overlapping clusters, each query could possibly belong to one or more clusters, thus forming relationships between clusters. For example, if a query belongs to two different clusters, then queries in one cluster are also related to those in the other cluster since they share a relationship with a common query. One can therefore imagine a large network of queries directly or indirectly related to each other. If a new information need expressed as a query can be matched with a cluster on this network, a user can navigate this network to explore what other people have searched for, thus harnessing the collective knowledge found there to meet this need.

Finally, maintenance of the query repository follows essentially the same process. This is a crucial function that will enable the QGV to harvest and recommend the latest queries that could possibly reflect new or changing information needs. In the current version of the QGV, maintenance is done periodically offline. Here, newly submitted queries are first compared with existing queries in the repository and incorporated into the query database only if they are unique. Otherwise, only frequency counts are updated. In both cases, the queries in the repository are then re-clustered.

Query Recommendation

When a user submits a query, that query needs to be matched to its closest cluster, and the queries in this cluster will serve as recommendations to the user. Here, a recursive algorithm is employed in which the submitted query, Q_i , is treated as the root node. The algorithm will first locate the query cluster $G(Q_i)$ containing the submitted query Q_i . This is based on comparing the query with all cluster centroids in the query repository and picking the nearest cluster. Given the previous definition of a query cluster, all its members are directly related to

Q_i and therefore, $G(Q_i)$ may be regarded as the first level in the network of queries related to the root node, as shown in Figure 7.

Figure 7. Locating related queries and their clusters.

Next, the algorithm will find clusters containing the members of $G(Q_i)$. For example since Q_1 is a member of the cluster $G(Q_i)$, the algorithm will locate the query cluster $G(Q_1)$, which forms the second level of nodes as shown in the figure. This process continues until a user-specified maximum level is reached. In the default implementation, this maximum value is five, which means only the first five levels of query clusters from the root node will be retrieved. Thus, the final set of queries related 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 . The set of all related queries is then packaged and sent to the QGV visualization interface for display and exploration.

Consider for example the query “data mining”. Suppose this query is matched to the cluster {“data mining”, “predictive data mining”, “knowledge discovery”, “data mining

applications”, ...}. This set of queries will form the level 1 nodes around the root (“data mining”). The algorithm will then expand all the individual members of this initial cluster to retrieve indirectly related queries. Suppose “knowledge discovery” is currently being expanded and results in the cluster {“data warehousing, OLAP, data mining”, “data mining journal”, ...}. These queries will form the level 2 nodes around the “knowledge discovery” node. This process continues until five levels of expansion (the default) are achieved. Figure 8 shows the partial query network.

Figure 8. Partial query network for “data mining”.

Evaluation

A qualitative evaluation on the QGV was performed to investigate its usability and usefulness in facilitating the user’s search process as well as to uncover areas for further improvement. Heuristic evaluation (Nielsen, 1994) was chosen because it is cost-effective, quick and easy to administer (Preece, Rogers and Sharp, 2002). The goal of heuristic evaluation is to find usability problems so that they can be attended to as part of an iterative design process. Six people participated in the evaluation of the QGV. All of the participants rated their computer

skills and ability to use search engines as satisfactory or above average. Most of them also indicated that they had been using search engines quite often.

Method

Participants were given a ten-minute introduction to the QGV followed by a practice session before carrying out two search tasks. 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”, while the second task was to “find information about a project which aims to represent mathematical documents in a form understandable by humans and interpretable by machines on the World Wide Web”. At the end of the evaluation, the participants were asked to complete a questionnaire about the QGV. Here, 19 questions were used to evaluate the interface and various features of the QGV. These were associated with the usability heuristics proposed by Nielsen (1994) and were measured along a 5-point Likert scale representing Strongly Disagree (1), Disagree (2), Neutral (3), Agree (4), and Strongly Agree (5). The remaining questions were open-ended and elicited participants’ opinions about the QGV, which included positive and negative aspects of the system.

Results

All participants were able to successfully complete the tasks, indicating the QGV’s usefulness in assisting users in helping them meet their information needs. With respect to usability, Table 1 shows the average score for each heuristic in which scores closer to 1 indicate strong non-conformance while scores closer to 5 indicate strong conformance. The table shows that most heuristics obtained a score of 4 or above indicating that participants found the QGV to be usable. The exception was the final heuristic “Help and documentation”. Here, two participants felt that the system could not be used unless

documentation was available. This is understandable given that novices to the QGV would need some time to become familiar with the system's features and operation given that the concept of graph navigation is relatively new and not often encountered on the Web.

Heuristic	Average Score (1 = Strongly Disagree, 5 = Strongly Agree)
Visibility of system status	4.0
Match between system and real world	4.1
User control and freedom	3.6
Consistency and standards	4.2
Error prevention	4.2
Recognition rather than recall	4.1
Flexibility and efficiency of use	4.5
Aesthetic and minimalist design	4.2
Help user recognize, diagnose and recover from errors	3.9
Help and documentation	2.3

Table 1. Summary of heuristic evaluation results.

Participants were also asked if they would consider using the QGV and whether they would recommend the system to other users if it were publicly available. All replied positively to these two questions. This response was encouraging and suggested the viability of such a system and the idea of collaborative querying. When asked about the positive features of the QGV, several participants commented that the system's visualization was very useful in helping them search for information since the recommended queries gave them more ideas on what terms to use. Participants also liked the options to control the visualization area. For example, one participant commented that the QGV was "really useful in speeding up the searching process and in better understanding the relationship between queries, since the weights between queries are clearly shown. It also made the 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”.

There were however negative comments about the QGV. Since many of the users were unfamiliar with graph navigation and terminology, most participants mentioned the lack of detailed documentation as a hindrance to the successful use of the system. This was due to the fact that some technical terms were used, such as “weight”, “locality zooming”, “root node”, etc. In addition, three participants complained that the system took a long time to execute. Given that the QGV client is a Java applet, delays are unavoidable as class files need to be downloaded to the Web browser prior to execution.

Conclusion

Collaborative querying is a promising approach to information retrieval that helps users formulate queries by harnessing the collective knowledge of other searchers. In the QGV, this collective knowledge comes from previously issued queries which are harvested and clustered to form query networks that represent related information needs. The queries in the network are explored in the QGV, helping users locate other queries that might help meet their current information need. An additional benefit is that our approach reduces the learning curve in using new IR systems since users can draw on “best search practices” and learn from the trials and tribulations of others in their quest for relevant and quality information.

The QGV differs from existing systems such as Eurekster which adopts a community-based approach to collaborative querying through search groups of users who share similar interests. Eureskster requires more effort on the part of the user in building the search group before the benefits of collaborative querying can be realized. In contrast, the QGV is non-

intrusive and operates by comparing the user's current query and existing clusters in the query repository in the background. The Community Search Assistant (Glance, 2001) is a Web-based collaborative querying system that mines transaction logs and uses the overlap of search result URLs as the similarity measure between queries. Similar to our approach, queries are then clustered based on their similarity values, and these are used as recommendations for users to refine their queries. Here, the QGV is differentiated in two aspects. Firstly we adopt a hybrid similarity measure that considers both the query terms and the documents in the search results listings during clustering. Secondly, we employ a graph-based visualization scheme for users to explore and interact with related queries unlike the HTML-based display used in the Community Search Assistant.

Work on the QGV is ongoing. Interface improvements in response to the preliminary evaluation will need to be done. This includes reducing the amount of technical terms used in the user interface and the introduction of help and documentation. In addition to the preliminary evaluation, more comprehensive user studies of the QGV will be carried out. In particular, we will compare the performance of the QGV against existing search engines in a range of search tasks and investigate the types of tasks in which the QGV will be most helpful to users. The incorporation of HTML-based interfaces will also be explored to address the problems of long download times, especially for computers with slow network connections.

Acknowledgements

This project is partially supported by Nanyang Technological University (NTU), Singapore research grant number RCC2/2003/SCI. We would also like to express our thanks to the NTU

library and the Centre for Information Technology Services at NTU for providing access to the queries for our experiments.

References

- Anick, P.G. (2003), "Using terminological feedback for web search refinement: A log-based study", *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 88-95.
- Beeferman, D. and Berger, A. (2000), "Agglomerative clustering of a search engine query log", *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 407-16.
- Belkin, N.J., Oddy, R.N., and Brooks, H. (1982), "ASK for information retrieval: Part I: Background and theory", *Journal of Documentation*, Vol. 38 No. 2, pp. 61-71.
- Billerbeck, B., Scholer, F., Williams, H.E., and Zobel J. (2003), "Query expansion using associated queries", *Proceedings of the 12th International Conference on Information and Knowledge Management*, pp. 2-9.
- Bruza, P.D., Dennis, S. (1997), "Query reformulation on the Internet: Empirical data and the Hyperindex search engine", *Proceedings of the RIAO 97 Conference*, pp. 488-99.
- Ellis, D. (1993), "A comparison of the information seeking patterns of researchers in the physical and social sciences", *Journal of Documentation*, Vol. 49 No. 4, pp. 356-69.
- Fu, L., Goh, D.H., and Foo, S. (2004), "Query clustering using a hybrid query similarity measure", *WSEAS Transactions on Computers*, Vol. 3 No. 3, pp. 700-5.
- Fu, L., Goh, D.H., Foo, S., and Na, J.C. (2003), "Collaborative querying through a hybrid query clustering approach", *Proceedings of the 6th International Conference on Asian Digital Libraries, Lecture Notes in Computer Science 2911*, pp. 111-22.

- Glance, N. S. (2001), "Community search assistant", *Proceedings of 6th ACM International Conference on Intelligent User Interfaces*, pp. 91-6
- Jansen, B.J., Spink, A., and Saracevic, T. (2000), "Real life, real users, and real needs: A study and analysis of user queries on the Web", *Information Processing and Management*, Vol. 36 No. 2, pp. 207-27.
- Kleinberg, J.M. (1998), "Authoritative sources in a hyperlinked environment", *Journal of the ACM*, Vol. 46 No. 5, pp. 604-32.
- Kovács, L., Micsik, A., and Pataki, B. (1999), "AQUA: Query visualization for the NCSTRL digital library", *Proceedings of the 4th ACM Conference on Digital Libraries*, pp. 230-1.
- Kraft, R., and Zien, J. (2004), "Mining anchor text for query refinement", *Proceedings of the 13th conference on World Wide Web*, pp. 666-74.
- Nielsen, J. (1994), "Enhancing the explanatory power of usability heuristics tools for design", *Proceedings of the SIGCHI 1994 Conference on Human factors in Computing Systems*, pp. 206-13.
- Page, L., and Brin, S. (1998), "The anatomy of a large-scale hypertextual web search engine", *Proceedings of the 7th International WWW Conference*, pp.107-17.
- Preece, J., Rogers, Y., and Sharp, H. (2002), *Interaction Design: Beyond Human-Computer Interaction*, John Wiley & Sons Ltd, New York.
- Raghavan, V.V., and Sever, H. (1995), "On the reuse of past optimal queries", *Proceedings of the 18th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 344-50.
- Roussinov, D., Tolle, K., Ramsey, M., McQuaid, M., and Chen, H. (1999), "Visualizing Internet search results with adaptive self-organizing maps", *Proceedings of the 22nd*

annual international ACM SIGIR conference on Research and development in information retrieval, pp. 336.

Salton, G. and McGill, M.J. (1983), *Introduction to Modern Information Retrieval*, McGraw-Hill, New York.

Setten, M.V., and Hadidy, F.M. (2000), “Collaborative search and retrieval: Finding information together”, available at: https://doc.telin.nl/dscgi/ds.py/Get/File-8269/GigaCE-ollaborative_Search_and_Retrieval_Finding_Information_Together.pdf

Smeaton, A., and van Rijsbergen, C. (1983), “The retrieval effects of query expansion on a feedback document retrieval system”, *The Computer Journal*, Vol. 26 No. 3, pp. 239-46.

Taylor, R. (1968), “Question-negotiation and information seeking in libraries”, *College and Research Libraries*, Vol. 29 No. 3, pp. 178-94.

Wen, J.R., Nie, J.Y., and Zhang, H.J. (2002), “Query clustering using user logs”, *ACM Transactions on Information Systems*, Vol. 20 No. 1, 59-81.

Wen, J.R., Nie, J.Y., and Zhang, H.J. (2001), “Query clustering using content words and user feedback”, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 442-3.