DESIGN AND DEVELOPMENT OF A QUERY GRAPH VISUALIZATION SYSTEM

Dion Hoe-Lian Goh¹, Chei Sian Lee², Alton Y.K. Chua³, Brendan Luyt⁴

¹Wee Kim Wee School of Communication & Information
Nanyang Technological University
31 Nanyang Link SCI Building
Singapore 637718
ashlgoh@ntu.edu.sg

²Wee Kim Wee School of Communication & Information
Nanyang Technological University
31 Nanyang Link SCI Building
Singapore 637718
leecs@ntu.edu.sg

³Wee Kim Wee School of Communication & Information
Nanyang Technological University
31 Nanyang Link SCI Building
Singapore 637718
altonchua@ntu.edu.sg

⁴Wee Kim Wee School of Communication & Information
Nanyang Technological University
31 Nanyang Link SCI Building
Singapore 637718
brendan@ntu.edu.sg

*Correspondence concerning this article should be addressed to Chei Sian Lee, Division of Information Studies, Wee Kim Wee School of Communication and Information, Nanyang Technological University, 31 Nanyang Link, Singapore 637718. Email: leecs@ntu.edu.sg. Phone: (65) 6790-6636. Fax: (65) 6791-5214
DESIGN AND DEVELOPMENT OF A QUERY GRAPH VISUALIZATION SYSTEM

Abstract

Collaborative querying is a technique that harnesses the collective search experiences of users to assist in query formulation. We present the Query Graph Visualizer (QGV), a visual collaborative querying system that recommends related queries to a user’s submitted query through a network visualization scheme. Users are able to explore the query network and select queries for execution on an information retrieval (IR) system. The QGV is not meant to be a replacement for IR systems but as a value-added module to help users search more effectively. Consequently, the QGV is not an IR system but works in tandem with existing IR systems. The design of the QGV is discussed, focusing on its architecture, the interaction between the QGV’s main components and the implementation of the user interface. An evaluation of the QGV was also conducted to assess the performance of the system against a conventional search engine. Results indicate that the evaluators who used the QGV completed their tasks much faster compared to those using a search engine alone. A usability evaluation also showed that the system complied with standard user interface heuristics.

Keywords: Collaborative querying, Information retrieval, Query graph visualize, Query network

I. Introduction

Search engines are arguably the most popular means of obtaining information on the Web. Despite their popularity however, there are several challenges faced by users of search engines and information retrieval (IR) systems in general. Firstly, users are often overwhelmed by the amount of information returned by search engines due to the explosive growth of the Web (Kobayashi & Takeda, 2000). Unfamiliarity with search engine syntax and options, or the domain being searched can also make it difficult for users to obtain relevant results (Lau & Goh, 2006). Another challenge for users is the vocabulary mismatch problem, or the failure to express an information need in terms compatible to those used in the system for representing potential information sources. For example, according to Nordlie (1999) in an analysis of online searches in a library setting, almost 50% of all failed searches are caused by vocabulary mismatch.

Much work has been done to improve the performance of IR systems. One area is in the algorithms used in indexing and retrieval. A second approach is through helping users refine or reformulate their queries
which can be accomplished via automatic query expansion or query recommending in which related queries are presented to users as alternatives to the original query. A possible implementation of query recommending is collaborative querying (Goh, Fu & Foo, 2005) in which recommendations are harvested from queries previously submitted by other users, thus harnessing the collective knowledge and search experiences of users in an IR system. Yet another way to improve the performance of IR systems is through the user interface. This can be done by visualizing search results or by visualizing query formulation (Shi, Irani & Li, 2007)

Information visualization helps people to deal with large amounts of information by taking advantage of our innate visual perception capabilities. In particular, graph visualizations have been used in many areas wherever there is an inherent relation among the data elements to be visualized (Herman, Melancon & Marshall, 2000). In IR, graph visualizations have been used to display the relationship between search results documents. For example, Börner (2000) demonstrated the organization of search results into clusters of semantically similar documents using Latent Semantic Analysis and the generation of a dynamic graphical visualization of the resulting categories and their documents in a two dimensional space.

At the same time, collaborative querying has been shown to be useful in helping users reformulate queries in IR systems (Fu, Goh, Foo, & Supangat, 2004). This approach assists query reformulation by recommending related queries. Since each query in turn can potentially receive query recommendations, a query network can be formed in which each query is connected either directly or indirectly to other queries via recommendations. From an implementation point of view, a query network can be presented as simply as a list of term recommendations or through a visualization system that renders the query network in a graphical form for exploration.

In this work, we combine the concept of collaborative querying and graph visualization, leveraging on the individual strengths of both techniques to help users better formulate their queries within an IR system. As discussed, the graph visualization technique is useful in exploring large datasets which contain inherent relations among the data elements. The data elements in this case are queries and the presentation of related queries in the graph visualization display is expected to help users find related queries to meet their information needs.
In our previous work (Goh, Fu & Foo, 2005), we have described the algorithms for generating query networks and have presented a visual collaborative querying system, the Query Graph Visualizer (QGV), for exploring these networks. Our focus in this paper will be on the implementation aspects of the QGV, focusing on its architecture, the interaction between the QGV’s main components and the user interface. We begin with a review of related work in this area, focusing on query reformulation and visual IR. We then describe the architecture and implementation of the QGV. Finally, a user evaluation of the QGV is described in which we present our results and analyses.

2. Related Work

The enormous amount of information available on the Internet and its exponential growth have made searching for relevant content a daunting task. Researchers have attempted to address this problem from different perspectives. This section focuses on specific techniques in query reformulation and user interfaces related to our work.

2.1 Query Reformulation

Query reformulation is an interactive process of query modification to help users meet their information needs. Automatic query expansion is one such technique and it attempts to improve the effectiveness of ranked retrieval by adding additional terms to the original query (Smeaton & van Rijsbergen, 1983). A query is run using conventional IR techniques and related terms are then extracted from the top-N documents in the search results listings using a variety of statistical heuristics. These terms are added to the original query and the expanded query is executed to generate a new set of documents. An example of a query reformulation system is Altavista Prisma (Anick, 2003) which parses the documents in the search results listings and uses the most frequently occurring terms as recommendations. In addition, Kraft and Zien (2004) proposed a technique to automatically expand a query by mining the anchor text of hyperlinks in HTML pages and using these are term suggestions.

Another query reformulation technique harnesses users’ search experiences or domain knowledge. Also known as collaborative querying, it is typically accomplished by analyzing the search history found in transaction logs maintained by the IR system. The queries are extracted and then used as recommended alternatives or as sources for automatic query expansion. The Community Search Assistant (Glance, 2001) is an example of a collaborative querying system that helps users formulate appropriate queries for a particular search session by showing relevant queries previously submitted by other users. In the
method proposed by Billerbeck, Scholer, Williams and Zobel (2003), 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 a performance increase of 26% to 29% over querying with no term expansion.

2.2 Visualization in Information Retrieval Systems

Studies have shown that users have problems with sifting through results that are simply presented as a long list of documents (Zamir & Etzioni, 1998) since these are not very intuitive for finding the most relevant documents in the result set. This has motivated work that explores the combination of IR with information visualization, and such systems are also known as visual information retrieval systems.

For example, Veerasamy and Heikes (1997) describe a visualization tool in which bars are used to represent retrieved documents, ranked by their relevance scores. Query terms are also represented by bars, with the height of the bar corresponding to the weight of that term in the document. Another example of a system that shows the relationship between queries and retrieved documents is AQUA (Kovács, Micsik, & Pataki, 1999). Applied to the NCSTRL (Network Computer Science Technical Reports Library) distributed digital library, the system visualizes query refinement as a series of panels. Each panel represents an operation used to generate the next set of results. Within each panel, users can choose from various options to construct a query. The panels thus not only show the query expression being generated but also provide feedback about the returned results.

3. QGV Architecture

An important consideration in the design of the QGV was that it is not meant to be a replacement for IR systems but as a value-added module to help users search more effectively. Consequently, the QGV is not an IR system but works in tandem with existing IR systems. Figure 1 shows a high-level overview of the QGV’s architecture. There are two distinct sections – the IR component and the QGV component. A user submits a query via an IR interface. The query is executed by the IR engine to retrieve a set of relevant documents. At the same time, the query is routed to the query network retrieval engine where the query network is generated. The network is then returned to the QGV where it is displayed. As the user interacts with the QGV, any query found in the network can be executed, repeating the entire process.
The architecture of the system follows a standard three-tier design and consists of the repository layer, applications layer and user interface layer. The repository layer consists of the documents and indexes maintained by the IR system and the query repository maintained by the QGV. The query repository is a relational database that stores the collection of previously submitted queries and their relationships. The application layer contains the IR and QGV logic. The former is dependent on the IR system through which the QGV interfaces with, while the latter consists of two components, both interacting with the query repository via an ODBC connection. The query clustering module is responsible for finding similar queries and grouping them into clusters. The network retrieval engine constructs a query network given a query as a root node and a set of network generation options. This is done by a recursive algorithm that first matches the root query with its cluster of similar queries, and then matching these other queries by their clusters, and so on, thus expanding the network into multiple levels. Details of the clustering and matching algorithms can be found in (Goh, Fu & Foo, 2005).

The user interface layer consists of the IR interface through which queries are submitted and search results documents are viewed, and the QGV interface in which query networks are explored. The QGV
interface communicates with its applications layer components via XML over HTTP. Here, XML is used to represent query network data to be rendered on the interface. An advantage of XML as the data format is that different visualizations of the query network are possible without any redesign of the applications layer components. For example, in the current version, the user interface is a Java applet that is responsible for rendering the query network. However, another implementation could offer a Scalable Vector Graphics version or a simple HTML interface using the XML data. The QGV interface also communicates with the IR interface so that refined queries can be executed. However, this is IR system specific and requires that appropriate application programming interfaces (APIs) be available.

4. QGV Implementation

The QGV is developed in Java due in part to its cross-platform support and the availability of a wide variety of class libraries for graphical user interface development, networking and XML processing. The following are the main Java class libraries used:

- **Java Swing.** Swing provides an extensive set of classes for building complex graphical user interfaces. The QGV interface is implemented as a Java applet and employs many of the standard components available in Swing. This includes frames, panels, buttons, labels, textboxes, and so on.

- **DOM (Document Object Model) and SAX (Simple API for XML).** The QGV uses XML for communicating query network data, and DOM and SAX are used for manipulating this XML-based data. DOM presents a XML document as a tree of elements in which data is represented as nodes and attributes. DOM interfaces provide a standardized and complete view of the XML document’s contents and support both access and manipulation by applications. SAX (http://www.saxproject.org) is a standard API that is useful for parsing XML documents and for converting XML documents into DOM-type objects.

- **JAXP (Java API for XML Processing) and Xerces Parser.** JAXP (http://java.sun.com/xml/jaxp/) enables applications to parse and transform XML documents independent of a particular XML processing implementation. JAXP implements the DOM and SAX standards and provides facilities to manipulate XML documents. Apache Xerces2 (http://xml.apache.org/xerces2-j/index.html) is used to parse XML documents and to convert them into DOM objects.

- **TouchGraph.** Query networks in the QGV are visualized using TouchGraph, an open source Java-based graph framework for presentation and navigation of networks.

(http://touchgraph.sourceforge.net/). TouchGraph provides useful API support for features such as mouse handling, customization of graph components, graph layout, zooming, etc.

### 4.1 The User Interface

The QGV’s user interface customizes and builds upon the graph functionality offered by the TouchGraph library. It is responsible for the following tasks:

1. **Query network display.** Rendering of nodes and edges, color settings, and network layout.
2. **Query network processing.** Requesting query network data from the server and parsing the network data received.
3. **User interaction handling.** Network navigation (zoom, pan and drag), triggering of content-sensitive menus, selection of query nodes, and submission of queries to search engine.

Figure 2 shows the QGV’s user interface. Each network node represents a 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 degree of relationship between these two nodes is 0.3. Relationship values are between 0.1 (least related) to 1 (most related).
In the QGV, a submitted query is known as the root node. Nodes that are directly connected to the root 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. The generation and presentation of a query network is governed by three parameters:

1. **Maximum number of queries per level.** This refers to the maximum number of nodes that will be generated at each level of the query network. For example, if this value is five, then every query node can have at most five child nodes connected to it.

2. **Maximum depth level.** This refers to the maximum number of levels the query network can have. Here, the root node is level 1, while all children connected directly to the root is level 2, and so on.
3. Minimum weight limit. This specifies the minimum allowable relationship value between two query nodes. Node pairs whose relationship values are lower than this are not presented on the query network.

Query networks can potentially be large and the QGV supports three navigation features for exploring them. These are:

1. **Zoom.** Zooming in and out of the query network is achieved by changing the length of the nodes’ edges. This feature is especially useful for looking at specific sections of large networks (see Figure 3).

   ![Figure 3. Before and after zooming.](image)

2. **Rotate.** This feature provides different viewing angles of the network by rotating all the query nodes using the root node as the pivot point. This feature is useful in situations when nodes overlap and a different viewing angle would rectify this issue (see Figure 4).

   ![Figure 4. Rotate feature.](image)
3. **Locality zooming.** This feature is used to control the number of levels to be displayed by expanding or collapsing child nodes into their respective parents. This is used to simplify network presentation in large query networks. Figure 5 shows an example of locality zooming. Note that the number beside each node denotes the number of child nodes that have not been expanded yet.
In addition, each query node is associated with a popup menu that provides options to submit the query to the IR system, make the query node the new root, and expand or collapse child nodes. Figure 6 shows that when a query is made the new root ("visual C programming"), a new query network is generated. Figure 7 shows how child nodes can be expanded. This feature is available when locality zooming causes child nodes to be collapsed into the parent. Note that the collapsing of child nodes is also available and is the reverse of Figure 7.
4.2 The Server

The server is responsible for generating data describing a query network using the contents of the query repository, given a request (the root node) from the QGV user interface. As described earlier, query networks generated by the server are represented in XML. Figure 8 shows the XML schema for query network data, while Figure 9 shows an example of a query network for the query "C++". Figure 5 shows its rendering on the QGV. The XML elements and their corresponding attributes are:

- **QueryContainer.** This is the root element and acts as the container for query network data.
  
  Its attributes are:

  - **QuerySearch.** Contains the submitted query and serves as the root of the query network.
  
  - **MaxQueryInLevel.** The maximum number of queries per level.
  
  - **WeightLimit.** The minimum weight limit between two query nodes.
  
  - **MaxDepthLevel.** The maximum depth level of the query network.
  
  - **UrlSearch.** The URL of the IR system to issue queries to.

- **Query.** This element contains information about individual queries in the network. Its attributes are:
  - **ID.** The unique ID of the query.
  - **Value.** The actual query string.
  - **Level.** The level of the query in the query network.
  - **Root.** Indicates whether the query is the root.

- **QueryLink.** This is the child element of the *Query* element and is used to indicate which queries are connected to a parent query defined by the *Query* element. Its attributes are:
  - **ID.** The unique ID of the child query.
  - **Weight.** The degree of relatedness between the child and parent queries.
Figure 8. XML schema for query network data.
5. QGV Evaluation

A pilot study was conducted to investigate the usefulness and usability of the QGV to facilitate IR. Participants were asked to complete two tasks using the QGV and a standard search engine for the purposes of accessing usefulness, after which they were asked to complete a usability questionnaire and provide feedback on the system.

5.1 Participants

Twelve participants served as evaluators. Of these, two were doctoral students in the information sciences, three worked in the engineering field, while seven were in the computing industry. In a self-report questionnaire, all evaluators rated their computer skills and ability to use search engines as satisfactory or above average. In addition, most had been using search engines quite often. Together, the responses suggest that the majority of the evaluators were reasonably skilled in conducting searches.

5.2 Methodology

The twelve evaluators were divided into two groups of six and were asked perform the same two search tasks using different IR systems. Group A used the QGV while Group B used the Google search engine. The time to complete each of the tasks successfully was recorded and served as a gauge of usefulness of the QGV.

Evaluators in Group A were provided with an introduction to the QGV and were given five minutes to explore the system before performing the search tasks. They were then asked to complete a questionnaire assessing the usability of the QGV. These questions were based on the usability heuristics proposed by Nielsen (1994). In addition, several open-ended questions were asked to determine the evaluators’ opinions about the QGV which included positive and negative features of the system. Unlike those in Group A, evaluators in Group B were not briefed about the QGV before doing their tasks. Instead, after completing their tasks in Google, they were shown the QGV and then given the opportunity to explore the system. They were then asked a few questions about their opinions of the system.
The tasks completed by all the evaluators were:

- **Task 1.** Evaluators had to find information about a Semantic Web Seminar held in 2002 which discussed various rule system techniques suitable for the Web. Information to be provided included title of seminar, its location, the host's and organizers' names, and motivation and goal of the seminar. Specific clues about the topic, which might be useful in constructing keywords for the search, were deliberately mentioned.

- **Task 2.** Evaluators had to find information about a project which aims to represent mathematical documents in a form understandable by humans and also interpretable by machines on the Web. Information to be provided included the project name, relevant mathematical and Web technologies used, and the name of prototype that is under development. No specific clues were given and evaluators were expected to fully rely on the systems' features to search for the requested information.

Finally, the heuristic evaluation technique was chosen to assess usability of the QGV because it is cost-effective, quick and easy to administer (Preece, Rogers & Sharp, 2007). In addition, Desurvire and Thomas (1993) found that heuristic evaluation not only predicted problems observed in laboratory studies but also encouraged evaluators to suggest improvements. However, it is known that heuristic evaluation is also subject to evaluators' bias and rather dependent on the skill of the evaluators (Preece, Rogers & Sharp, 2007). With this concern, we ensured that the evaluators' backgrounds were consistent with the goals of the study in that all were information technology-literate and were comfortable with using search engines.

### 5.3 Results and Analyses

The two search tasks had varying levels of difficulty. Task 1 was more well-defined due to the clues given which could be used as query terms, while Task 2 was more open-ended due to the lack of clues. Table 1 shows the average time (and standard deviation) needed by each group to complete the two search tasks. The results suggest that evaluators in both groups took a slightly longer time to complete Task 2 as compared to Task 1.

More importantly, evaluators in Group A took far less time to complete both tasks (approximately 3 minutes each) as compared to those in Group B (20 minutes for Task 1 and 23 minutes for Task 2). The standard deviation for Group B was also greater than Group A. Taken together, the results suggest that the QGV consistently helped in the search process for each of the tasks. Note however that due to the

small sample size, our findings cannot be generalized although this initial study shows promise in the usefulness of the QGV.

<table>
<thead>
<tr>
<th>Task</th>
<th>Group A (QGV)</th>
<th>Group B (Google)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SD</td>
</tr>
<tr>
<td>1</td>
<td>2.92</td>
<td>1.59</td>
</tr>
<tr>
<td>2</td>
<td>2.94</td>
<td>1.02</td>
</tr>
</tbody>
</table>

Table 1. Task completion times – mean and standard deviation.

Nielsen’s (1994) ten heuristics were used to assess usability. In the evaluation, each heuristic is represented by one or more questions concerning the design of the QGV. Each question is rated along a five-point scale indicating the level of agreement – “strongly disagree”, “disagree”, “neutral”, “agree” and “strong agree”. Table 2 reports the average number of participants who rated “agree” or “strongly agree” to a heuristic (which may consist of multiple questions). The maximum value per heuristic is six as only evaluators in Group A performed the evaluation.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility of system status</td>
<td>5</td>
</tr>
<tr>
<td>Match between system and real world</td>
<td>5</td>
</tr>
<tr>
<td>User control and freedom</td>
<td>4.8</td>
</tr>
<tr>
<td>Consistency and standards</td>
<td>5.5</td>
</tr>
<tr>
<td>Error prevention</td>
<td>5.5</td>
</tr>
<tr>
<td>Recognition rather than recall</td>
<td>4.8</td>
</tr>
<tr>
<td>Flexibility and efficiency of use</td>
<td>6</td>
</tr>
<tr>
<td>Aesthetic and minimalist design</td>
<td>5.3</td>
</tr>
<tr>
<td>Help users recognize, diagnose and recover from errors</td>
<td>4</td>
</tr>
<tr>
<td>Help and documentation</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. Scores for heuristic evaluation of the QGV.
As shown, nine of the heuristics scored 4 or higher by the evaluators suggesting that the QGV was considered a usable system. The only heuristic that the QGV did not appear to comply with was “Help and documentation”. This however was not surprising as the system was new to the evaluators who would require time to become familiar with its operation. In addition, the concept of network navigation was also new and evaluators would have to be acquainted with terms such as “nodes”, “edges”, “network depth”, and so on. Nevertheless, the fact that evaluators could still complete their assigned tasks in a much shorter time than those using Google was encouraging and suggests the viability of the system in helping users meet their information needs.

In addition, evaluators were asked for their opinions on the QGV. Of the six who used the QGV to complete their search tasks, five felt that the system helped speed the search process while one remained neutral to the idea. When asked whether they would use the QGV if it were made publicly available, all 12 evaluators responded positively and would also recommend the system to other users. This overwhelming response was encouraging given the fact that of the 12 evaluators, six used the QGV for the search tasks while the other six only explored the system for a few minutes upon completion of their search tasks.

Qualitative comments about the positive and negative aspects of the QGV were also elicited. The positive features of the system can be summarized as follows:

- **Visualization scheme and user interface.** Several evaluators commented that the visualization was useful in helping to search for information from the Web. Other evaluators mentioned that the use of color, for example to denote weights of links, was useful. One evaluator who had experience with Java commented that the user interface was very responsive.

- **Graph options.** Most of the evaluators mentioned that the zooming and rotation features as well as options to specify the minimum weights between queries were able to help them better understand the relationship between queries.

On the other hand, the negative features of the QGV include:

- **Slow loading time.** Most of the evaluators remarked that the system took a long time to load. This could be due to the size of the Java applet which had to be fully downloaded before execution could begin.

- **Need for documentation.** Although evaluators were briefed on the use of the QGV, the need for documentation was still expressed. This was in part due to the use of technical terms as well as the fact that most evaluators were not familiar with the concept of network navigation.

6. Conclusion

Collaborative querying is a promising approach to information retrieval that helps users formulate queries by harnessing the collective knowledge of other searchers. This paper describes the design and implementation of the QGV, a visual collaborative querying system. The QGV is designed to help users formulate queries to an IR system through the exploration of query networks, which are rendered graphically on the QGV’s user interface. An evaluation of the system suggests the viability of using it to support information retrieval. In particular, evaluators were found to complete their search tasks faster using the QGV and in general, they expressed agreement that the system complies with Nielsen’s ten usability heuristics.

Although the evaluation has yielded useful results, generalizations cannot be made as only 12 participants were involved. A larger scale evaluation involving more users and more tasks is therefore necessary to ensure validity and reliability of our results. In addition, providing documentation is a necessary next step to ensure that users of the QGV are familiar with the terminology and features of the system.

Acknowledgements

This project is partially supported by NTU research grant number RG25/05.

7. References


