Query Graph Visualizer: A Visual Collaborative Querying System

Dion Hoe-Lian Goh, Alton Y.K. Chua, Chei Sian Lee, Brendan Luyt
Wee Kim Wee School of Communication and Information, Nanyang Technological University
{ashlgoh, altonchua, leecs, brendan}@ntu.edu.sg

Abstract

Collaborative querying harnesses the collective search experiences of users for 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 design of the QGV is discussed, focusing on its architecture 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.

1. Introduction

Search engines are a popular means of obtaining information on the Web. 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 [11]. Thus, it becomes increasingly difficult for users to locate relevant information. Another challenge is the vocabulary mismatch problem – the failure to express information needs in terms compatible to those used in the system for representing potential information sources. For example, in an analysis of online searches in a library setting, almost 50% of all failed searches are caused by vocabulary mismatch [14].

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 is through helping users refine or reformulate their queries. This can be accomplished via automatic query expansion [20] or query recommending [9] in which related queries are presented to users as alternatives to the original query. A possible implementation of query recommending is collaborative querying [8] 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 performance is through the user interface – by visualizing search results or by visualizing query formulation [17].

Information visualization enables people to deal with large amounts of information by taking advantage of our innate visual perception capabilities. In particular, graph visualizations are often used whenever there is an inherent relation among the data elements to be visualized [10]. In IR, graph visualizations have been used to display the relationship between search results documents. For example, [4] 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. Another example is Fetucino which provides a user-interface for visualization of search results including advanced graph layout and display of structural information [2].

At the same time, collaborative querying has been shown to be useful in helping users reformulate queries for IR systems [8]. As described, this approach assists users in reformulating their queries 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.

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 previous work [7, 8], 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. The present paper will instead 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, 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.

2. Related 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 example and it attempts to improve the effectiveness of ranked retrieval by adding additional terms to the original query [18]. A query is run using conventional IR techniques and related terms are then extracted from the top-N documents in the 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. Examples include HiB [5] and Altavista Prisma [1]. In addition, [12] proposed a technique to automatically expand a query by mining the anchor text of hyperlinks in Web pages. The process involves examining the original query, finding all anchor texts that are similar and presenting these as query term suggestions.

Another query reformulation technique harnesses users’ search experiences. Also known as collaborative querying, it is typically accomplished by analyzing the search history found in transaction logs maintained by IR systems [8]. The queries are extracted and used as recommended alternatives or as sources for automatic query expansion. The Community Search Assistant [9] is an example that assists in query formulation by showing relevant queries previously submitted by other users. In the method proposed by [3], 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 IR Systems

Studies have shown that users have problems sifting through results that are simply presented as a long list of documents [21] 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, or visual IR systems [17].

For example, [19] describes 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. In the VIBE (Visualization by Example) system, users are able to define reference points called Points-of-Interest (POI) [15]. Each POI, represented by a circular icon, is positioned on a 2-D display. Each document, represented by a square icon, is positioned in the display based on how it relates to the POI. VIBE is also able to visualize documents retrieved by a series of queries on a single display, with the POIs being the different queries.

3. QGV Architecture

An important consideration in the design of the QGV was that it was 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 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. 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 [7].

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. XML is used to represent query network data to be rendered on the interface. An advantage of XML is that different visualizations of the query network are possible without any redesign of the applications layer components. 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.

Figure 1. QGV high-level architecture.

4. QGV Implementation

The QGV is a Java applet and the following are the main Java class libraries used:
- Java Swing provides an extensive set of classes for building complex graphical user interfaces.
- JAXP (Java API for XML Processing) (https://jaxp.dev.java.net/) 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.
- Query networks 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 clicks, customization of graph components, graph layout, zooming.

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), 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 becomes the root node. Nodes 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 especially useful for looking at specific sections of large networks.

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.

3. Locality zooming. This 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.

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.

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. 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.

5. QGV Evaluation
A pilot study was conducted to investigate the usefulness and usability of the QGV to facilitate IR. 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. The twelve evaluators were divided into two groups of six and were asked to 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 [13]. 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. In contrast, evaluators in Group B completed their tasks in Google. They were then shown the QGV and given the opportunity to explore the system. They were also 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. Clues 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.

Finally, the heuristic evaluation technique was chosen to assess usability of the QGV because it is cost-effective, quick and easy to administer [16]. In addition, [6] 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 [16]. 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.

5.1. 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. The results thus suggest that the QGV consistently helped in the search process for each task. However, due to the small sample size, our findings cannot be generalized although this initial study shows promise in the usefulness of the QGV.

Table 1. Task completion times.

<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>

Nielsen’s [13] 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.

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 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.

**Table 2. Heuristic evaluation scores of the QGV.**

<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>

In addition, evaluators were asked for their opinions on the QGV. Of the six who used the QGV, 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.

**6. Conclusion**

This paper describes the design and implementation of the QGV, a collaborative querying system 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 necessary to ensure validity 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.

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

**7. References**


