Interactive Visualization of Search Results of Large Document Sets

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

James D. Anderson
B.S.C.E., Wright State University, 2016

2018
Wright State University

Wright State University
GRADUATE SCHOOL

October 31, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY James D. Anderson ENTITLED Interactive Visualization of Search Results of Large Document Sets BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

Thomas Wischgoll, Ph.D.
Thesis Director

Mateen M. Rizki, Ph.D.
Chair, Department of
Computer Science and Engineering

Committee on
Final Examination

Advisor, Thomas Wiscgholl, Ph.D.

Michael Raymer, Ph.D.

John C. Gallagher, Ph.D.

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Anderson, James D. M.S.C.E., Department of Computer Science and Engineering, Wright State University, 2018. *Interactive Visualization of Search Results of Large Document Sets* .

When presented with many search results, finding information or patterns within the data poses a challenge. This thesis presents the design, implementation and evaluation of a visualization enabling users to browse through voluminous information and comprehend the data. Implemented with the JavaScript library Data Driven Documents (D3), the visualization represents the search as clusters of similar documents grouped into bubbles with the contents depicted as word-clouds. Highly interactive features such as touch gestures and intuitive menu actions allow for expeditious exploration of the search results. Other features include drag-and-drop functionality for articles among bubbles, merging nodes, and refining the search by selecting specific terms or articles to receive more similar results. A user study consisting of a survey questionnaire and user tracking data demonstrated that in comparison to a standard text-browser for viewing search results, the visualization performs commensurate or better on most metrics.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to take this opportunity to extend my thanks to the Human Centered Big Data (HCBD) research group. In particular, Brad Minnery and Michael Raymer provided expert insight and Eric Nichols implemented a semantic hashing search which became invaluable to this project.

I'd also like to thank my advisor Thomas Wischgoll whose guidance and encouragement through the process were invaluable. Additionally, I am grateful to all his students for their moral support and positive comments of my work.

Of course, I'd be remiss if I did not mention my awesome parents, Alan and Judith, whose support has allowed me to peruse my continued academic interests.

Dedicated to

Mike Bostock

Introduction

Searching large document sets quickly and efficiently presents a challenge to data analysts who may or may not have a precise set of search terms capable of generating the specific results for which they are seeking. Analysts may have millions of documents at their disposal and only a few search terms in mind, and that search query can return thousands or more results. The analyst may desire to query a broad topic area and filter out undesired results, focusing on various sub-topics present within the results. Additionally, the applications available to browse search results are mostly pure text-based giving a listing of results which may or may not be ranked. If the results are ranked, there may be documents deep in the list that may give insight into the search; however, since the results are lower in the list, the likelihood they will be seen is less than results higher in the list.

In contrast to the typical textual listing of results, graphical search browsers offer a different approach to presenting search results. Graphical browsers typically provide a visual representation with similar results grouped closer together, and the groupings can be represented with text summaries, whether it be the encompassing topics or terms shared by the documents. The visual representations also allow users to explore and interact with the results in novel ways not available with traditional search browsers.

This paper seeks to validate the hypothesis that a visualization of search results facilitates a quicker discovery of desired information when compared to a traditional text-based approach. To perform this analysis, a graphical browser was designed and implemented, and a user study was conducted to validate this premise. The system design visualizes the

document set in a tree structure with the main search terms represented at the root and the more refined results appearing in child nodes. The project is developed as a web-based application which utilizes the built-in interactivity that a web-browser provides as well as being cross-platform compatible. The evaluation was conducted with participants performing a search task and afterward providing feedback on the application. Other metrics, including task time and tallies of items viewed, were recorded and interpreted to assess the benefit and suitability of the application.

This paper is outlined with the following main points: (1) the introduction to the problem, (2) a presentation of available search methods and visualizations of document sets, (3) a description of the methods and implementation details to execute the design, (4) a presentation of the results from evaluation of the system, and finally (5) a discussion of the results.

1.1 Theory

In terms of theory, this project attempts to address several research goals associated with visualizing text-based information. First, it is posited that the visualization assists a user with browsing large data-sets. Second, the visualization aids a user seeking particular information without a clear search query for the desired results. Third, the interactive features of the visualization aid the user in browsing the results and visualization. These theories are tested via a user study involving survey questions to gather feedback on the effectiveness of the graphical browser.

1.1.1 Large Document Sets

The main theory guiding the design of the visualization is when presented with a large set of documents retrieved from a search, an interactive and graphical representation of the

document set will assist the user in browsing the search results. Visually grouping similar results enables the user to quickly choose documents within a desired topic area. After exploring documents surrounding a certain topic, the user may wish to refine the search to get results more specific to that topic. Thus, the visualization will help provide an overview of the search results, as well as facilitate additional searches and refinement of the results.

1.1.2 Inadequate Search Query

In addition to aiding in the search process, another theory involving the visualization is if a user is unsure of how to form a query to obtain the desired information, the ability to interact with the results graphically will provide the user with a variety of visual cues and shortcuts to facilitate a speedier search. There are numerous ways the user may be enabled to develop the search.

Aside from the search results being displayed in the traditional text method, clusters of documents are represented as a word-cloud, as seen in a sample of the graphical browser design in Figure 1.1. The terms in these word-clouds can provide inspiration for additional search terms which may provide results that appeared with a lower ranking or not at all in the original search. Additionally, the user may desire to conduct multiple searches in parallel. The visualization provides the ability to transfer results from one search to another, which may also inspire new insight in the process.

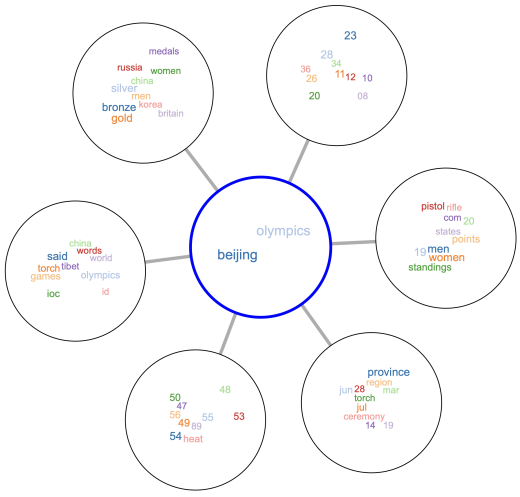

Figure 1.1: Search results in the designed graphical browser, word-clouds represent clusters of similar documents

1.1.3 Increased Interactivity

A final theory involves the interactive capabilities of the visualization. Aside from when the user types the initial search terms, the visualization is controlled with either a mouse or touch display interface. All actions and modes are available from on-screen menus. The selection of articles and search terms for refining the search are all performed by clicking or dragging the desired label. These interactive capabilities allow the user to navigate the data in a much simpler manner as opposed to using keyboard shortcuts to perform various actions.

1.2 Scope

The scope of this thesis is the design, implementation, and evaluation of an interactive visualization for search results. The design process included research into existing methods of interaction with search results, techniques into visualizing large document sets, and experiments with various software packages to accomplish these design goals.

The data set utilized for the underlying search consists of text documents. The application is capable of displaying search results from a web-based search engine; however, the visualization methods cover only the text content of the web pages. Therefore, the scope of this project only encompasses text-based search results and not the images, video, or other media that may appear on the web pages.

To evaluate the visualization, a study of use cases was conducted with user tracking data and questionnaire feedback collected to compare the efficacy of a graphic search versus a traditional text-based browser. The evaluation covers the suitability of the application for browsing through large search results, and each user undertook the task twice: once with the visualization and once with a standard text-browser. This approach allowed for participants to compare the ease, speed, and effectiveness of both browsing methods.

Outside of the scope is the development of a new visualization library, as existing

data visualization libraries exist and provide adequate functionality to implement the design. While the design of the visualization utilizes dynamically generated word-clouds, the development of algorithms for the generation of word-clouds is outside the project's scope. Also outside of the scope is the development of new search algorithms. While part of the project involves parsing and clustering search results, these techniques are utilized on results from existing search methods.

Background

Users wishing to browse search results are typically given a plain text listing of the documents returned from the query. With the advent of better graphics capabilities and algorithms to find patterns among documents, visualizations have developed as an alternative. To meet this end, clustering processes are often utilized to group similar documents. Additionally, the structure of many data sets from the inter-connectivity of websites and the links between concepts and categories contained within ontologies, can be visualized as a graph, there has been much work on visualizing these large relational data structures.

2.1 Traditional Search

Traditionally, search results of text-based data are viewed in a list format. Shown in Figure 2.1 is the Google search browser, which in addition to the results listed in text, provides a summary of the topic. Typically the results are shown sorted by a ranking metric such as relevance to the search query or by publish date. For example, Google uses a PageRank [8] calculation to order the results. In general terms, PageRank weights a website by summing the PageRank

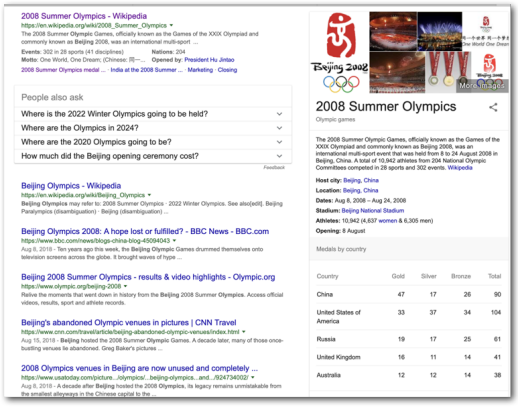

Figure 2.1: Google Search Results

values of other sites containing links to the page. The results of the search query are listed using the title of the web-page. For Google, the anchor text, or the click-able portion of a link, for each search result is not just the text of the web page but also any images, video, programs, or databases.

2.2 Document Set Visualizations

To visualize textual information, various methods have been developed to reduce the complexity from hundreds and thousands of terms to a less sizable and more human-understandable space. Two common methods, document topic generation and clustering described in [9] and [10], use term frequency-inverse document frequency (TFIDF) with word-vectors and Latent Dirichlet allocation (LDA). TFIDF is a statistical method of representing a document set as a numerical matrix. Each document occupies a row of the matrix, and each term indexes into a column. The weight of each term is proportional to the frequency with which the term appears within a particular document and inversely proportional to the number of documents in which the word occurs. A typical way of calculating TFIDF is

$$TFIDF_{ij} = \frac{f_{ij}}{max_k f_{jk}} \log \frac{N}{n_i}$$

as illustrated in [11], which normalizes the frequency of each term.

LDA is a statistical model which assigns probabilities to topics based on the collection of terms within a document [12]. With LDA, connections between documents can be made even though the documents themselves may consist of mostly disjoint sets of terms.

K-Means is a clustering algorithm which determines k centroids where each element is assigned the label of the closest centroid. A variant on K-Means, Mini Batch K-Means [13] is frequently used to reduce computational time while producing results very close to what the original K-Means generates.

Once terms and topics are generated from the documents, a common visualization method is a word-cloud. With this technique, the terms are displayed, typically arranged as to minimize empty space, with their font sizes proportional to the significance or probability of being related to the data set. Rolled-out Wordles [2] demonstrates a heuristic for building word clouds by removing overlaps between elements. In [1] several methods are presented, including one called TagCrowd which represents documents as word-clouds with the size of a word being proportional to its frequency within the text. Wordles and TagCrowd are shown in Figure 2.2.

Figure 2.2: TagCrowd [1] (top) and Wordle [2] (bottom)

While much work has been done creating algorithms to generate word clouds, there is some disagreement on the benefit to visualizing information in word clouds, notably in Jacob Harris' article [14]. A criticism is the semantic information contained in word clouds offer only a rudimentary view of the information. With the increased prevalence of using word clouds journalistically, Harris contends that the narrative of a news story is lost with overuse of word clouds to give readers a quick summary of an article.

Much work has been done on visualizing texts and document sets. Various projects in the topic of text visualization include using word frequency to generate a visualization and devising topics for either a single document or a set of multiple documents. Some visualizations deal with graphing word frequencies as a function of time, such as Google's n-gram viewer [3], shown in Figure 2.3 which allows the user to search

Figure 2.3: Google Ngram Viewer [3]

for multiple terms and compare their usage over time.

Another technique called Hierarchical point placement (HiPP) [1], shown in Figure 2.4 has circles, or "bubbles", with proximities proportional to similarity between the document sets, while the circles represent similar documents. DiTop-View [4], a visualization method with bubbles and word-clouds, partitions the canvas into different background colors which represent major topic areas, as seen in Figure 2.5.

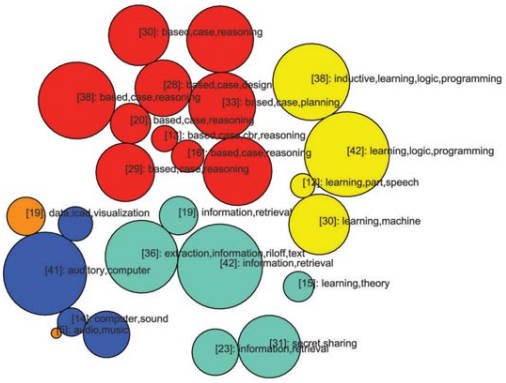

Figure 2.4: Hierarchical point placement [1]

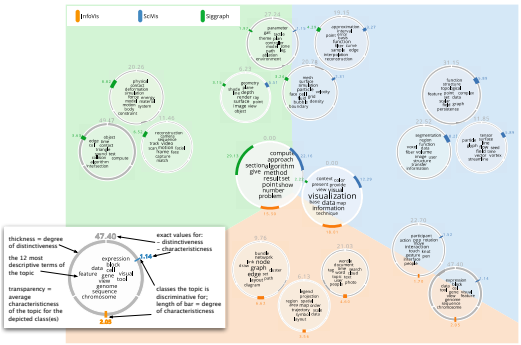

Figure 2.5: DiTop View [4]

In an approach different from the ones mentioned above, [15] enables users to view various graphical representations of the data including histograms, box-plots, and scatterplots. These features can then be broken down to easily understandable one or two dimensional displays, allowing for a better understanding of the higher level, multidimensional data. Another visualization pertaining to multidimensional data is Parallel Sets [16] which connects categories to each other using parallelograms, and the user can interactively move these parallelograms to remap connections among categories.

2.2.1 Clustering Visualizations

Many visualization methods utilize document clustering to group semantically similar documents. One such, iVisClustering [5], clusters documents by topic utilizing LDA to generate a graph visualization where closely related documents are grouped together with a

display of topic words, as shown in in Figure 2.6.

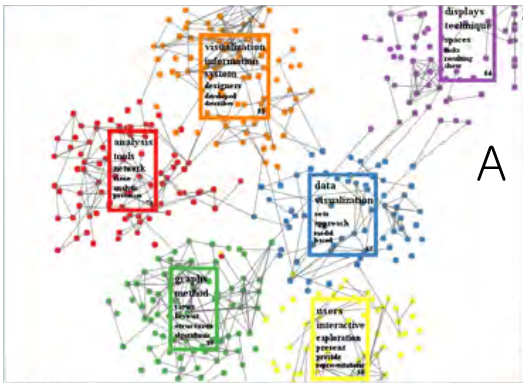

Figure 2.6: iVisClustering [5]

Another method, XCluSim [17], has various overview modes for displaying clustered data, one of which utilizes a force-directed layout with the document similarity visualized by the distance in between the nodes. Other methods include Radial Sets [18] to visualize set memberships for a large number of elements with overlaps among the sets and TIARA [19] which uses horizontal layers to separate topics. In contrast, the method used in [20] visualizes features into vertical stripes. With Termite [21], terms and topics appear in a tabular layout where the size of the circle is how well the term corresponds to a topic.

In [22], a system for information visualization is described that transforms the text into a spacial representation that still preserves the semantic meaning without language processing, thus reducing the analyst's mental workload. The result is a either a 2- or 3-dimensional visualization with the documents as points, where the proximities of documents are proportional to their similarities, allowing the analyst to quickly browse among similar documents.

2.3 Graph Visualizations

Graphical frameworks for depicting relational data have been developed for visualizing large-scale ontologies. Ontologies are formal representations of concepts, categories, objects, and data, as well as the relationships among them. The design of the search browser detailed in this paper takes inspiration from ontology-related visualizations. In particular, WebVOWL [6], shown in Figure 2.7 is a web-based visualization tool for graphic display

of an ontology. WebVOWL utilizes Scalable Vector Graphics (SVG) code and Cascading Style Sheets (CSS) presentation along with the JavaScript library Data Driven Documents (D3) [23] to display force-directed graphs. A force-directed graph is a visualization method for generating positions of nodes in a graph by simulating the vertexes as having an electric charge which will repel the other vertexes and the edges geometrically constraining the nodes. This approach allows for dynamic addition, removal, and repositioning of nodes, as the visualization will adjust to the change in graph structure.

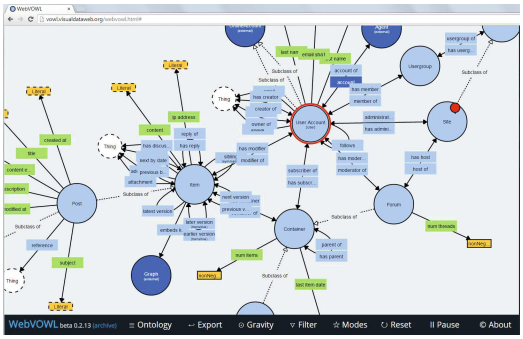

Figure 2.7: WebVOWL Ontology Visualization [6]

The specification for the ontology input interface is detailed in the VOWL 2 [24] standard for visualizing ontologies. In particular, circles in the visualization represent classes, solid lines represent properties with arrow heads depicting the direction the properties are applied. Rectangular boxes represent data-types and property labels. Colors also have specific meaning, with various colors representing the various functionality of the element.

Another inspiration for this project is the TouchGraph Navigator [7], shown in Figure 2.8. Similar to WebVOWL, TouchGraph can create visualization for the web; however, it is implemented in Java. TouchGraph allows the user to import data tables which are then visualized in a graph structure. It contains clustering algorithms which will reveal relations intrinsic to the data. Additionally, the application can visualize the interconnectivity of web sites on the Internet by graphing the links between pages.

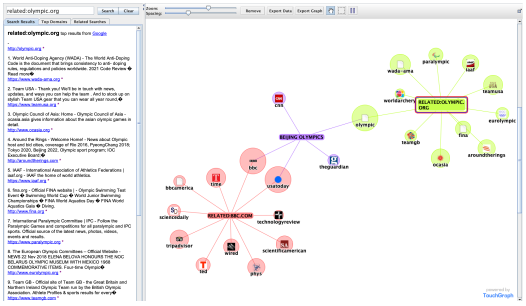

Figure 2.8: TouchGraph Navigator [7]

The above approaches have the advantage of providing the user with a high-level overview of the documents, objects, or concepts being visualized. Further, they are capable of illuminating patterns hidden within the dataset. However, several of the visualizations provide only a static display of the results and do not allow the user to manipulate the visualization to customize the search.

In contrast, the approach presented in this paper allows users to perform actions on the visualization, such as merging groups of results to refine the search into a particular topic. The user may select a term in order to view for documents more associated with the desired term. Furthermore, the user can select articles to receive additional results similar to the chosen documents.

Additionally, determining an adequate set of search terms for the query can be a challenge. For example, when searching Google Scholar for "*search visualization*," there are several topics under which the papers will occur. Articles about searching within visualizations appear as well as visualizations of networks and the inter-linkings of the Internet. Given these results, the visualization presented here allows the user to select topics relating to the desired search, view additional search terms to facilitate a better search, and filter out unwanted results. From the related works discussed above, there currently exists no single solution capable of providing all the above functions.

Implementation

The visualization system designed allows users to get a general overview of the topics their search terms may cover and then narrow down the scope of the search until discovering desired results. After the user enters the initial query, the application generates a central "bubble" that contains the search terms. This bubble acts at the root for a tree in which each child represents a subset of documents of its parent. In each child node, a word-cloud depicts the most prevalent topics and terms from the set of documents it represents. Figure 3.1 depicts an overview of the visualization.

In terms of functionality, the user can refine a search by selecting a term in one of the bubbles, and a new child is created to represent the documents which best fit this term. After multiple children have been created from a single parent, those children can be merged together to represent a new subset of documents. Children can be merged by performing a union operation on the document sets.

With regards to search data, the system draws its input from a machine learning-based search. This search algorithm utilizes a neural network and semantic hashing [25]. For this project, a dataset of Reuters articles serves as the basis for search queries. The visualization, running in a web-browser, makes requests to a custom search script hosted on a server. The script executes the search and returns the results formatted for the visualization.

Because of the built-in graphics capabilities and interactivity of most modern web-browsers, the visualization is implemented with JavaScript. This allows the code to be

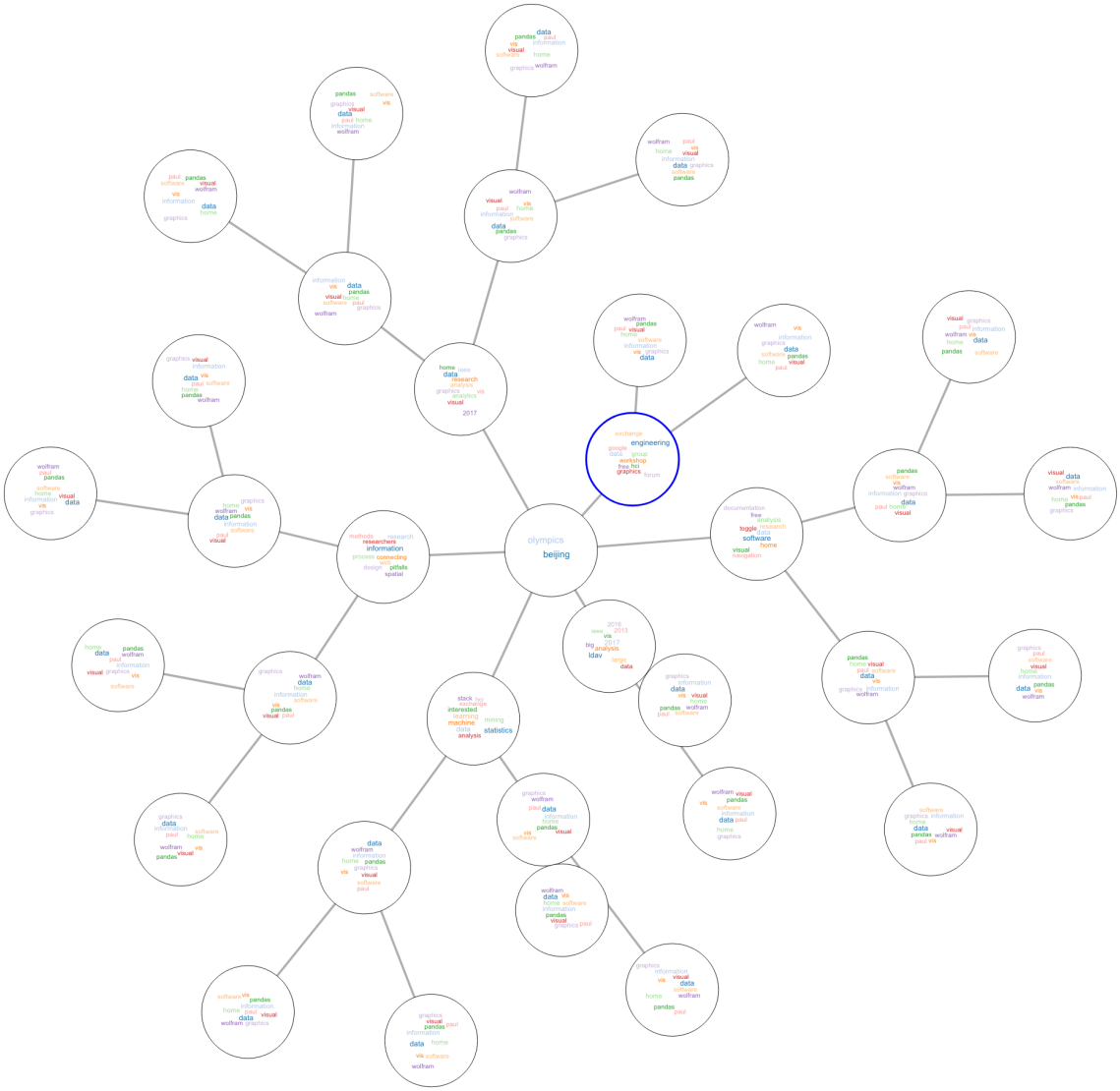

Figure 3.1: Search Graph Visualization

extremely portable. For the purposes of this project, Firefox 60.0.2 (64-bit) [26] was utilized for development; however, the code is known to work with various browsers such as Google Chrome and Safari.

3.1 Client-side Visualization

The main visualization provides an interactive, graphical display of the search results. As seen in Figure 3.2, the visualization is divided into three sections: input (top-left), output (bottom-left), and the graphical tree (right). The division between the left and right panels is maintained by the Split.js utility [27], which enables the divider to be slid, growing or shrinking the areas to each side.

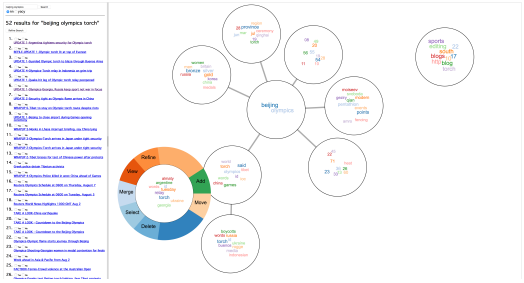

Figure 3.2: Visualization Screen Image

In the input area, a standard text-input box allows the user to type in the initial search query. Below the input area is the text-based output of the search results. As seen in Figure 3.3 these appear as hyperlinks, enabling the user to easily access the results. With each result ”Yes” and ”No” check-boxes enable the user to indicate whether they wish to see results similar or dissimilar to the given document as part of the *Refine* action.

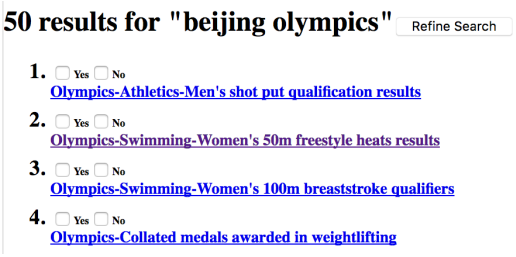

Figure 3.3: Example of Results with Check-boxes

The right section of Figure 3.2 contains the visualization of the search results. When the program begins, a single bubble with a word-cloud containing the words *suggested search terms bubble* appears in the upper-right. As the user clicks on documents, this

bubble is populated with terms relating to that document. When the user initiates a search by typing terms into the text-input, a bubble will appear in the visualization area containing those search terms. This bubble is the root of the new search query. A search query request is initiated, and when the response is received, new bubbles are generated connected to the root, and they are populated with word-clouds containing terms related to a sub-group of the entire search.

3.1.1 Main Search Visualization

The structure of a search result is presented visually as a force-directed graph utilizing the D3 library for JavaScript. D3 provides a programming interface through which Hypertext Markup Language (HTML) elements such as SVG can be manipulated. D3 also provides layouts to visualize datasets. The layouts automatically generate the appropriate geometry for the provided data which then can be utilized to render the visualization. This project utilizes two D3 layouts: force graphs and pie charts.

Force Graphs

The force layout simulates charged particles that are constrained by the links between nodes in order to generate the positions of the nodes [28]. As such, the charge strength for each node can be set determining how strongly each node repels each other, as well as the strength of the links between connected nodes. Additionally, a friction attribute determines how quickly the nodes' velocity decays.

The parameters for the force-directed graph need to be tuned according to the size of the bubbles such that the nodes are an appropriate distance from each other and they respond suitably to being moved by the user. The default parameters are set for relatively small node sizes with a radius value around 10 pixels. To accommodate text inside the circles, the nodes in the visualization are assigned a radius of 200. As such, the parame-

ters need to be increased, as summarized in Table 3.1. Additionally, once the user begins interacting with the graph, the friction parameter is reduced (which increases the effective friction) from 0.9 to 0.5 in order to prevent the nodes from drifting excessively.

Charge	Friction	Link Distance	Charge Distance	Link Strength
-30000	0.9/0.5	50	750	1

Table 3.1: Force-directed Graph Parameters

To create a force graph layout, a list of nodes and links must be provided. Nodes are typically implemented as a JavaScript object with various key and value pairs. Nodes are given *x* and *y* attributes from the force layout. The links between nodes are mappings of *source* and *destination* nodes. The link connects the center of the *source* node with the center of the *destination* node. These must be updated as the node positions are recalculated, which can be attached to the *tick()* function of the force layout.

In order to have the nodes drawn in the browser, a graphic element must be appended. This element can be any graphic, but for simplicity an SVG circle is used to represent the search nodes. Once attached, D3 provides functions to alter the HTML attributes of SVG elements. This functionality is typically performed in the form of:

```
element.attr("attribute", value)
```

Listing 3.1: D3 Attribute Example

where *"attribute"* is the desired parameter to be set. For SVG circles, some attributes include radius, stroke (color of the outline) and stroke width. The *value* provided can either be a constant or a function which returns a value based on the node. The function that gets called is given two parameters: the node data and the node index. This function gets called for each node, providing an alternative to looping through each node.

Once the initial state for the nodes is defined, they can be modified by using *d3.select()* or *d3.selectAll()*. These functions either select a specific element given it's unique identifier or a group of elements based on their class identifier. Using these selections, the graphic elements can be dynamically manipulated. For example, in this project the nodes are frozen after they are dragged to a position. Otherwise the dragged node will drift away. Once the user begins dragging another node, the previous node becomes unfrozen. This functionality is implemented simply as:

```
d3.selectAll('#' + bubble.node_id).classed("fixed", bubble.fixed = true)
```

Listing 3.2: D3 Freeze Node Example

where *bubble* is the node to be frozen. Using the pound-sign signifies the node is being selected by ID instead of class.

Along with providing the details for HTML graphic elements, D3 allows event listeners to be attached to SVG elements. For example, functions can be written on events such as *mouseover, mouseout, mousedown,* and *mouseup*. There are also events specific to touch screen devices which this project utilizes. When an event such as a mouse click is triggered, the click is registered for all elements the mouse is currently over. This may be undesired, since each bubble is being drawn into the browser window, and when clicking a node the event is triggered for both the node and the window. To avoid this, D3 provides a function *d3.event.sourceEvent.stopPropagation()*. When this function is called, any other elements involved in the event will not have their event listener called. This is particularly useful in the word-cloud function, discussed in 3.1.1. Additionally, the default actions that occur on these events can be overridden by using *d3.event.sourceEvent.preventDefault()*. This function is called when the nodes are being dragged. Instead of the default event, D3 provides custom functionality for handling drag events.

Word-clouds

To summarize the results contained within a node, the visualization utilizes word-cloud representations. The concept behind the word-cloud representation is to provide a quick overview of a group of search results as well as allowing the user suggestions on additional terms which may be helpful in refining the search query. Visually, the font size of each term corresponds to how strongly the term correlates to the set of documents contained within that bubble. The font size is relative to a given bubble and not to the search results as a whole.

The visualization treats the word-cloud in the suggested search bubble differently than the word-clouds in the rest of the visualization. When the user views a search result, terms related to that result are placed in the suggested search term bubble. These terms can be dragged into a bubble in the main search result space providing additional terms to refine the search query.

The word-cloud layout is generated using a D3 layout called *d3.layout.cloud()* [29]. A list of words mapped to sizes is passed to the layout generator, and an object containing functions, including mappings of x- and y-coordinates for each word, is returned. This word-cloud layout object can be used as the data to render the SVG text elements into the web browser. The calculation for the coordinates for each word is generated by the Wordle algorithm developed by Jonathan Feinberg [30]. In summary, this algorithm places the first word near the center of the word-cloud canvas, and each successive word is placed into the canvas such that it does not collide with any other words. If the word does collide, it's position is stepped along a spiral starting at the center of the canvas.

One modification made to the word-cloud layout generation for this project is to change the layout from fitting the words into a square area. Since the word-clouds for this project reside within circular bubbles, the word-cloud positions are bounded to a circular layout. This modification may be useful in future iteration of the program, such as changing the bubbles from circles to ellipses. In this case, the major- and minor-axes attributes

can be passed into the word-cloud layout generator.

3.1.2 Interface

The visualization provides several means for the user to interact with and refine the search results. The application has been programmed to allow for both a mouse and multi-touch displays to be utilized. There are two categories of interactions: gesture actions and menu selections.

Gestures

Listed in Table 3.2 is a summary of available gestures. For the purposes of this project, gestures pertain to both touch and mouse pointer actions. Much of the functionality of the mouse is copied for touch functionality, but some actions are handled separately. For instance, using the mouse-wheel will zoom-in and zoom-out of the visualization while the same action is achieved with a pinch-gesture on a touch display. Navigating the entire visualization is achieved by clicking or touching on a blank area and dragging the canvas.

Function	Detail	Mode/ Action	Local/ Remote
Click word	If in Add mode, narrows the search by including the given word	Action	Remote
Drag word	Words from the ”suggested bubble” can be dragged into the search	Action	Local
Check Yes/No	Upvote or downvote the document to refine the search	Action	Local
Drag link	Drag link from one bubble to another; words associated with the document are added to the new bubble	Action	Local

Table 3.2: Descriptions of Gestures

Several objects in the visualization can be dragged around the screen. When dragging a bubble with the mouse pointer or touch display, the attached bubbles will follow

and reorient themselves, allowing the user to rearrange the configuration of the visualization. When using a multi-touch display, multiple bubbles can be dragged simultaneously, including those which belong to the same search or bubbles of a separate search. Other items capable of being dragged include the terms in the *suggested search term bubble*. The user can drag-and-drop these terms into any of the search bubbles, allowing a new term to be utilized in a search refinement.

A final set of items capable of being dragged are the document links. The user can drag a link from the left-panel into one of the search bubbles. Links can be dragged to any search bubble with the result depending on if the destination bubble is or is not the same as the source bubble. If the destination is the source, then the link is simply marked "Yes." If the source and destination are different, the document is added to that bubble (if not already there) and marked as "Yes." Additionally, terms that are associated with the document are added into the bubble's word-cloud. In this way, the bubble is updated to reflect the additional document.

In order to view the documents contained in a bubble, the user can either hover the mouse over the bubble or touch and hold. After doing so, the contents will appear in the left-pane of the web-browser. Additionally, the menu will appear around the bubble.

Menu

The menu interface provides access to the various functions which can be performed. Table 3.3 details these functions. Most of the menu items describe actions that take place immediately when the button is clicked or pressed. However, two of the items, *Add* and *Move* change the mode of interaction. When the application is in *Move* mode, bubbles can be freely moved and the ability to click terms is disabled. When the mode changes to *Add*, the bubble terms become click-enabled and doing so will cause the application to perform an additional search using that term, and the query result is added to the visualization.

Function	Detail	Mode/ Action	Local/ Remove
New Search	Send search terms to server and add received nodes	Action	Remote
Add	Enables ability to click words	Mode	Remote
Merge	Combine terms & documents in selected nodes	Action	Local
Delete	Remove node and reconnect its children to parent	Action	Local
Select	Highlight bubble and add to selected list (or un-highlight and remove from list)	Action	Local
View	Loads the documents into the left-panel to view the results	Action	Local
Refine	Takes the up- and -down votes & uses them to refine the search	Action	Remote

Table 3.3: Descriptions of Menu Items

The rendering of the menu is done utilizing D3's pie chart layout and SVG arcs. The menu is shown in detail in Figure 3.4. To generate a layout, the menu data is given to *d3.layout.pie()*, which returns a layout for the pie chart. Parameters such as start and end angle can be given to specify where the layout should begin and end. For this project, the pie chart is generated as a complete circle. The pie layout is then passed as the data for an SVG arc, where the menu is actually rendered. For the SVG arc, the inner and outer radius can be specified. This is utilized to create a cut-out for the search result bubble, as the center of the pie chart is translated to the x- and y-coordinates of the node.

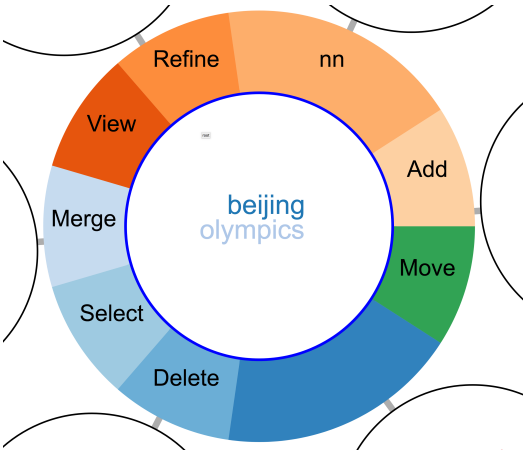

Figure 3.4: Menu Detail

With reference to the interactivity of the menu, touch and mouse events are handled differently. Normally, the menu is hidden, and to show the menu the user can either hover the mouse over or touch-and-hold the bubble. This action also reveals the search results

for that particular node in the left-side pane of the window. When the mouse moves out of the menu, the menu is removed; however, the results will stay in the left pane. To remove the menu with a touch screen, the user can touch on an empty space of the visualization. Additionally, if the user hovers or touches a different node, a new menu will appear and the existing menu will be removed.

3.2 Architecture

The system consists of two main parts: the client-side visualization and the server-side search routine. The client-side application implemented in JavaScript utilizes the D3 data visualization package. When the user enters search terms for a new search, the application instantiates an HyperText Transfer Protocol (HTTP) request to the server. The server parses this request and imparts the search to a semantic hashing neural network.

3.2.1 Server Search Request

When requesting data from the server, the client can set various parameters, which are detailed in Table 3.4. A few parameters include the search query terms, the number of documents to be returned, the number of word-clouds to be generated, and the minimum and maximum font sizes for the word-clouds.

3.2.2 Client Data Format

The method in which the design is implemented enables the client to visualize search results from a variety of sources as long as the search results are parsed into the appropriate format. After sending a search request, the client expects the response to be a JavaScript Object Notation (JSON) string. The JSON object contains two parts: *nodes* and *results*. The

Parameter	Description
q	Search query terms
n_arts	Number of results to be returned
n	Number of nodes to be generated
stype	Search engine to be used
q_type	Indicates whether query is new/initial or refinement of existing search
yes_inds	List of articles marked ”yes” for refinement
no_inds	List of articles marked ”no” for refinement
font_max	Maximum font size for word-clouds
font_min	Minimum font size for word-clouds
topics	Boolean to generate topics for ”suggested bubble”
font_topics	Font size for ”suggested bubble”
debug	Boolean to print debug info
cached_search	Boolean to return saved results

Table 3.4: CGI Parameters

system also recognizes a third field, *topics*, which optionally can be utilized to populate a suggested search term space.

The *nodes* field is a list of data used to populate the bubbles. Each object in *nodes* contains the following fields: *ids*, *titles*, *radius*, *name*, *words*, *sizes*, *top_arts*, *word_vectors*, *links*, and *descriptions*. A detailed explanation of what each field represents is depicted in Table 3.5

The other main field, *results*, contains the cumulative results for the search query, which contains lists for *titles*, *links*, *ids*, and *descriptions*. For searches where one cluster is returned, these fields are simply duplicates of the single node.

To initiate a search, the JavaScript sends the parameters in the Uniform Resource Locater (URL) string. These parameters are detailed in 3.3.1.

3.3 Server-side Search

In order to demonstrate the visualization’s ability to present search results from an active search, a simple back-end is developed to generate results, parse the results for the visu-

Field	Description
ids	List of article/document IDs
titles	List of titles/headlines for each document
radius	List of radii for each bubble
name	List of names for each cluster
words	Lists of terms making up word-clouds
sizes	Lists of font sizes parallel to words
top_arts	For each term in words, an ordered list of document ids most associated with the term
word_vectors	For each document, a mapping of {word: value}
links	List of HTML links for each document
description	List of descriptions for each document

Table 3.5: Table of Fields in *nodes*

alization to display, and return the parsed data. When the user enters a search query or chooses to refine the search parameters, an HTTP request is sent to a server hosing a script. This script runs the search utilizing a neural network trained to generate search results from a set of Reuters news articles. The script then clusters the results, forming the basis of the nodes for the visualization.

3.3.1 Apache CGI

To run the search and parse the results for visualization, a Python script is hosted on an Apache web server with Common Gateway Interface (CGI) enabled. To access this script, the client side visualization generates an HTTP request and asynchronously receives the results. The main benefit of this approach is the data for the visualization can be parsed remotely, freeing up the client machine to continue processing the visualization. Additionally, utilizing Python takes advantage of the numerous libraries available for doing text analytics. Finally, since the visualization application relies on a remote process to generate the data to be visualized, the application is able to connect to other search engines depending on user needs.

To generate data suitable for the visualization, the Python script obtains search results via a neural net search discussed briefly in section 3.3.2. The results of the search consist

of Reuters article texts and headlines as strings. The text of each article is used to cluster the documents into similar groups using a K-Means Mini-Batch process [13]. The inputs to the clustering algorithm are word-vectors generated by TFIDF value for the term compared to the other terms in the given word-cloud.

Along with generating terms for the word-clouds, the script is capable of generating topic terms for the search as a whole. To accomplish this, the TFIDF vectors for each document are used to calculate the non-negative matrix factorization (NMF) [31] for the document set. NMF is capable of extracting topics from a document set, and these topics can be utilized by the visualization.

3.3.2 Semantic Hashing

Although the scope of this project focuses on the visualization of search results and not the development of search algorithms, a brief discussion of the search process utilized can aid in some of the understanding of parts of the visualization. Semantic hashing [25] involves training a neural network with the inputs being documents represented as word-vectors where at each index is the frequency of a particular term. The output is a bit-vector; for this project a 32-bit vector is utilized. The distance between the bit-vectors of two different documents represents the semantic similarity between the two texts.

To obtain a set of results from the network, an input vector is generated utilizing the search terms in the query. The network outputs a single 32-bit vector, and the articles most similar to this vector are returned. Because the similarity of the documents are measured by distance between their bit-vectors, the results can be ranked in comparison to the bit-vector obtained from the search query.

In addition to returning results from a search query, semantic hashing also offers the capability to generate results similar to a given set of specific documents. Thus, semantic hashing provides the ability to refine search results. After determining the documents most similar to the initial search, the bit-vectors of specific documents can be compared with

vectors of other documents to find more articles semantically similar. The visualization incorporates this function in two ways: 1) the user can check the ”Yes” box next to specific results and 2) the user can drag a document into a bubble. The latter action has the effect of checking the document as a ”Yes” as well as populating the bubble with terms related to the dragged article. After performing either of these actions, the user may choose to ”refine” the search. Doing so executes the above described capability of finding documents similar to a subset, and these results are returned to the visualization as a refinement of the search.

Evaluation

Various metrics to evaluate visualizations exist, including monitoring system performance values such as frame rate, CPU workload, and memory utilization. Additionally, researchers will perform user studies to judge the visualization's effectiveness at providing insight and understanding of the data. In [32], participants were given a specific task to complete inside a virtual reality environment. A user study was performed to test the time and accuracy of the participants.

Studies involving the efficacy of various search browsers have been done, specifically in [33] where users were presented with a visualization with half the participants given an option to search the contents and the other half explored the data with no search option. The metrics used to guage the use of the visualization included *intent*, or whether the user sought specific data within the visualization and *active search count* or the number of data items viewed. The study in [33] also analyzed timing data of the participants, including *exploration time* and *average visit time during exploration*.

There has also been work on the benefits of visualization for understanding document sets. One such, PolicyLine [34], focuses on visualizing sets to aid in political decision making efforts. PolicyLine offers analysts with a graphical pipeline taking document texts as its database. The system was evaluated by having participants perform specific tasks followed by a brief questionnaire of the overall system.

In terms of the number of participants for the study to have viable statistical significance, various factors influence the appropriate sample size. For example, the goal of the

evaluation as well as the approach toward assessing the results can be factored into sample size. In [35], Forsell suggests at least 12-14 participants. Additionally, Carpendale states that qualitative evaluations typically require smaller sample sizes than quantitative studies [36]. While the tracking data involves quantitative data on user performance, the questionnaire presents mostly qualitative feedback on the visualization.

The design of the evaluation of the graphical search involved having participants perform a search task in the graphical browser as well as a text browser. After the task, the participants answered survey questions to give feedback on the search browser. Additionally, tracking data such as amount of time to complete the task and a count of the number of articles viewed were tracked during the search task.

4.1 Study Design

To evaluate the application, a group of volunteer participants ($N = 12$) were asked to perform a task utilizing the graphical visualization search browser and asked to complete a user survey afterwards. In order to remove the variable of the effectiveness of the semantic hashing neural network search, for each participant the results from the same search query of ”beijing olympics” were given. Within these results, the user was then tasked with finding a distinct piece of information; each specific task is listed in Table 4.1. The query and tasks were chosen based on the data-set for which the neural-network was trained. The data utilized as the search basis was a set of 94,065 Reuters articles from the time-frame of around 2007-2008.

All participants performed the search-task on the same machine running Firefox in Windows 10. A touchscreen display was utilized, and users were given the option to browse employing either touch- or mouse-gestures, or a combination of both input modes.

Search Task	Sample Response
Where the Olympic flame started in China	Hong Kong
How many days Olympics torch relay went around the world	130
Nickname of Olympic stadium in Beijing, Chine	Bird's Nest
Greek stadium where Olympic torch hand-off occurred	Panathinaio
US city on Olympic torch route	San Francisco
Group protesting torch tour in China	pro-Tibet

Table 4.1: User Study Search Tasks

The surveys were designed to test the effectiveness and convenience of the visualization versus a normal text-based search browser. In conducting the surveys, the user was asked to conduct a simple search task. In order to facilitate a comparison between the two browsing methods, all survey participants performed the same task twice: once with the graphical-browser and once with the text-browser shown in Figure 4.1. Half of the participants utilized the graphical-browser first and the other half performed the task first in the text-browser. In the presentation of results that follows, the former group is named Group 1 and the latter named Group 2.

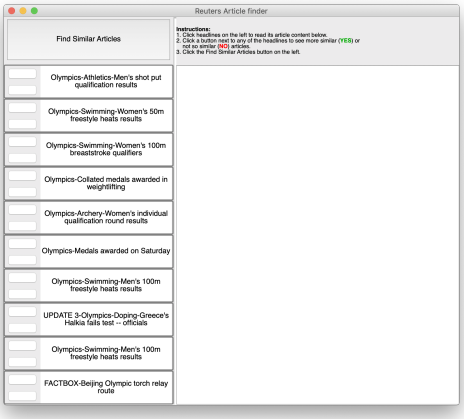

Figure 4.1: Text Browser for User Study

Another metric collected to evaluate the application was user tracking data. While performing the task, the system recorded user tracking data of mouse and touchscreen activity. Every action taken by the user was logged with a time-stamp, allowing timing data to be extrapolated.

4.2 User Surveys

After completing the search task on both the graphical and text browser, participants completed a survey comprised of multiple choice questions and one open-ended question. Some questions were designed to gauge whether the user felt the graphical browser or text-based display were better for performing various parts of the task. Other questions asked how well the visualization performed with respect to showing the overall structure of a set of search results. Some questions sought feedback on the various features of the application. The open-ended question asked for suggestions on how to improve the visualization.

A summary of the questionnaire statistical analysis is shown in Table 4.2. The listing shows the means of responses from Groups 1 and 2, as well as the overall mean, normalized between 1 and 0. Responses of 1 are most favorable to the visualization and responses of 0 are least favorable. Also shown are the standard deviations (σ) for all responses to the particular question. These values range from 0.15 to 0.31. A t-test was done for each question looking at the difference in responses between Groups 1 and 2, and while one question received a p-value of 0.051, the other p-values were relatively larger. Because of this, a power analysis was done ($\alpha = 0.05, power = 0.8$) to determine a suitable sample size to validate the differences between the groups. A few questions indicate a sample size of around 35-40 would be sufficient; however, several of the required sample size values suggest a much larger sample size is required. This result may also indicate there is actually no significant difference between the two groups, and the participants viewed utilizing the graphical browser equivalent to the text browser. If the sample size resulting from the power analysis is greater than 100, those values in the table are shown as greater than 100 to represent this fact.

Survey Question	Mean Group 1	Mean Group 2	Mean Total	σ	p-value	Required Sample Size
In general, describe using the visual graphical browser compared to the standard text-based browser	0.7500	0.6250	0.6875	0.1884	0.0510	36
Do you think the graphical search features allowed you to perform the task more quickly?	0.6667	0.5417	0.6042	0.2491	0.1066	63
How convenient were the graphical search features compared to text-based searching?	0.7381	0.7143	0.7262	0.1548	0.2998	$>$100
Overall, how would you rate the graphical search in terms of showing an overview of the results?	0.5417	0.7083	0.6250	0.2261	0.1520	29
How effective did you find the menus that would appear around the search bubbles?	0.4167	0.3750	0.3958	0.2491	0.3604	$>$100
How useful was the ability to select a search term in a bubble to refine the search?	0.7083	0.7917	0.7500	0.2132	0.5000	$>$100
Overall, how difficult was it to perform the task?	0.7917	0.6667	0.7292	0.2251	0.2998	51
How would you rate the overall performance of the graphical search?	0.6250	0.7083	0.6667	0.1628	0.2998	60

Table 4.2: Summary of Survey Statistics

4.2.1 Graphical vs. Text Browser

Looking as the survey responses, participants generally viewed the visualization to perform at least on par with a text-based search browser. In Figure 4.2, when comparing the graphical- versus text-based browser, a majority of participants responded they either somewhat or a great deal liked the visualization over the standard text representation. All participants responded they thought the graphical search features were at least as convenient as the text-based approach, detailed in Figure 4.3. While one respondent strongly disagreed that the graphical search allowed the task to be performed more quickly than the text-based browser, the rest of the responses felt the task was performed at least as quickly or better. Figure 4.4 depicts the user responses related to the convenience of the two search modes.

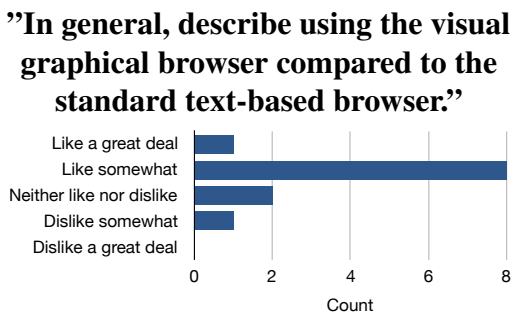

Figure 4.2: User Survey Responses to Question 2.2

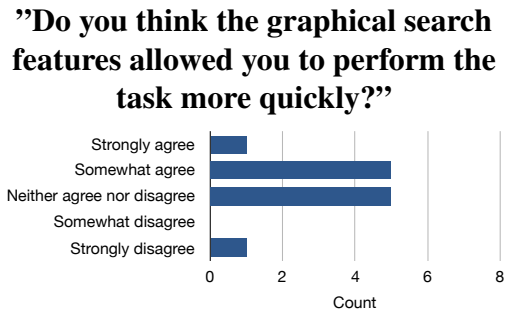

Figure 4.3: User Survey Responses to Question 4.3

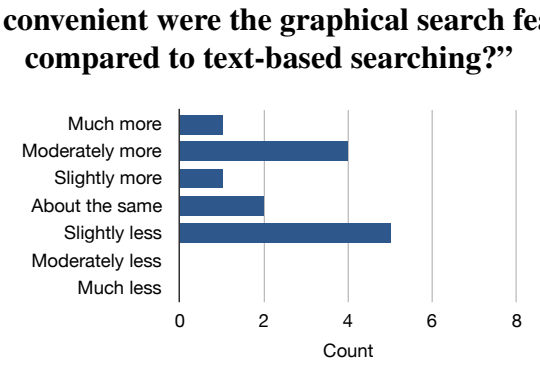

Figure 4.4: User Survey Responses to Question 4.1

4.2.2 Visual Representation of Results

In terms of the effectiveness of displaying an overview of the search results, Figure 4.5, shows that most participants found the graphical search very effective. When asked which method would be easier to browse through large amounts of search results, Figure 4.6 details that participants were closely split between the standard text-browser and the graphical browser .

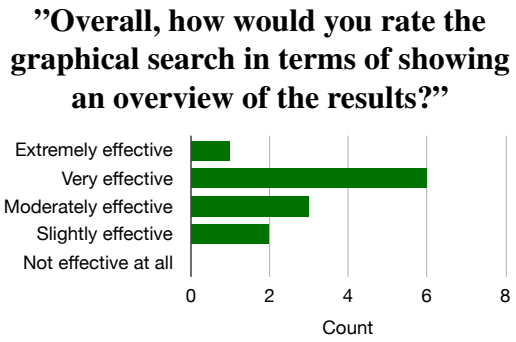

Figure 4.5: User Survey Responses to Question 4.5

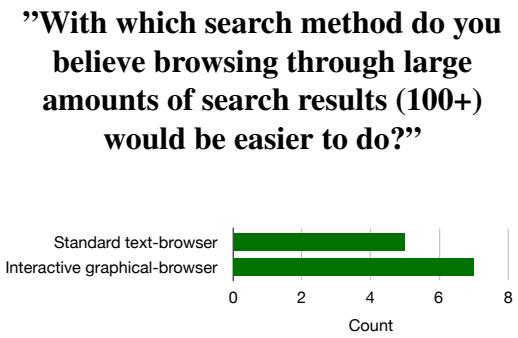

Figure 4.6: User Survey Responses to Question 2.3

4.2.3 Visualization Features

When asking about the features of the visualization, participants responded mostly positively about specific capabilities of the visualization. Participants were given the choice of input mode (touch versus mouse), and Figure 4.7 reveals that roughly two-thirds of users utilized the touch screen. Of those who responded they employed the touch screen, Figure 4.8 relates that two-thirds of participants felt the touch screen made browsing somewhat easier where the other third felt it either made browsing somewhat harder or had no effect. Figure 4.9 illustrates the sur-

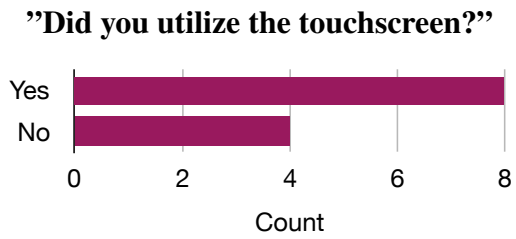

Figure 4.7: User Survey Responses to Question 3.2

vey responses regarding the effectiveness of the menus. No users responded that the menu was extremely effective. Most respondents stated the menu was either very or moderately effective. The rest felt the menus were either only slightly effective or not effective at all.

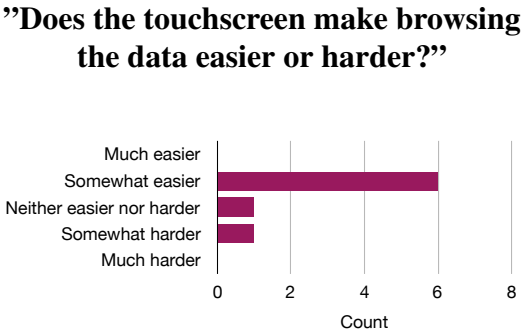

Figure 4.8: User Survey Responses to Question 3.3

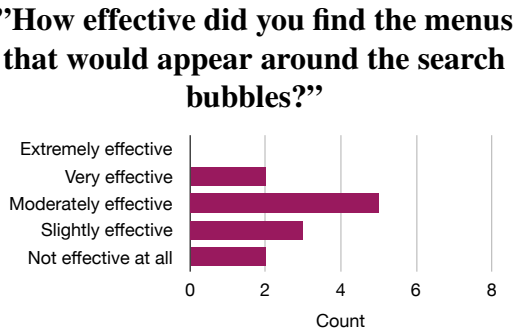

Figure 4.9: User Survey Responses to Question 3.5

4.2.4 Open-ended Responses

One open-ended question was asked in the survey: *In what way would you improve the search?*. Out of the 12 participants, 9 provided a response to this question. These 9 feedback statements generally relate into 3 different components of the application: Interactivity, the Visualization, and the Search Engine. Specific responses are listed in Table 4.3. In terms of interactivity, responses generally stated a preference for more gesture based interaction. Responses about the visualization varied, from suggesting making the size of the bubbles correspond to the number of articles represented to being able to focus on a particular search term. One statement regarded the formatting of the article text, which was presented as unformatted American Standard Code for Information Interchange (ASCII) text. Since the data was provided as plain text, the articles were displayed with no processing. Both the graphical browser and text-based search displayed the articles in this way, mostly to remove any bias with respect to either browsing mode. The last category of responses related to the search engine itself. One regarded the fact that some bubbles con-

tained primarily numbers, and the other recommended more training of the neural network.

In what way would you improve the search?

Interactivity	*"polish interactivity and intuitive movements"*
	"I do not like holding the link to recieve [sic] results"
	"simple gestures rather than the menu"
Visualization	*"Made it easier to locate the article I was looking for."*
	"Make the size of the bubbles correspond to the number of articles found in each category. Also, include more training for the person doing the search."
	"Be able to single in on one particular search term, or to further refine a particular group."
	"Formating the text when drilling down."
Search Engine	*"Some of the bubbles had primarily only numbers, which weren't useful in finding information. So more words, or words relating to those numbers."*
	"I would retrain the NN because a lot of the scripts were similar"

Table 4.3: Survey Text Responses

4.3 User Tracking

Participants were placed into one of two groups; Group 1 utilized the graphical browser first and Group 2 performed the task with the text-browser first. Because of this arrangement, the user tracking data figures are presented with participant timing information organized by these groups. Group 1 data is displayed in blue and Group 2 data is shown in red. Statistics which combine Groups 1 and 2 are presented in yellow. The x-axis of the figures contain the groups and the y-axis shows the value of the specific metric. The individual user within a group is placed in random order, but the ordering from chart to chart remains the same. At the right of the tracking charts is a summary of the group data with the y-axis scaling remaining the same as the individual data.

The tracking results from the study indicate a trend toward the visualization allowing users to discover the desired information in less time when they have no prior knowledge of which article contains the desired information. Table 4.4 displays a summary of the

statistical evaluation of the tracking data with relation to the performance of users in Group 1 compared to Group 2. For each of the metrics, a t-test reveals that a statistical difference in the performance of each group most likely does not exist. Since the p-values for the total time to perform the task and the time viewing the articles are both around 0.17, a power analysis was conducted to determine an adequate sample size to have enough data to make the results statistically significant. Since the p-values for the number of articles and bubbles viewed were relatively large ($p \gg 0.05$), it is surmised there is no statistical significance between the groups with respect to these metrics.

Metric	Mean Group 1	Mean Group 2	Mean Total	σ	p-value	Required Sample Size
Total Time (s)	374	539	456	253	0.173	38
Time Viewing Articles (s)	197	330	263	205	0.170	38
Number Articles Viewed	3.0	4.2	3.6	2.88	0.272	91
Number Bubbles Viewed	11.0	11.2	11.2	7.08	0.484	>100

Table 4.4: Summary of Tracking Statistics

4.3.1 Total Task Time

In terms of comparing the time the user took to complete the tasks in the graphical browser versus the text browser, Table 4.5 summarizes the statistics of these measurements. In general, users were able to complete the task in half the time utilizing the text browser versus the graphical. A paired t-test on the data resulted in a p-value of 0.016, which is less than the traditional threshold of 0.05. A power analysis shows that the required sample size would be 12, which is what the sample size for this study was. With these results, it can be said with some certainty that users will perform the task quicker with the text browser. One factor possibly contributing to these results may be the difference in the number of results each mode was capable of displaying. The text-browser utilized was only capable of displaying 10 results; however, the graphical-browser presented the user with 50 results.

Metric	Mean Time (Graphic Browser)	Mean Time (Text Browser)	σ	p-value	Required Sample Size
Total Time (s)	456	179	238	0.016	12

Table 4.5: Task Times with Graphical vs. Text Browser

Figure 4.10 displays the total amount of time it took for each user to complete the task. The participant who completed the task in the shortest amount of time was in Group 2 at 68 seconds (1:08 min:sec) as well as the individual who took the longest to finish at 924 seconds (15:24 min:sec). The mean time for completion of all users was approximately 456 seconds (7:36 min:sec). The average time to finish the task for individuals in the 1st group was 374 seconds (6:14 min:sec) and for participants in Group 2 it was 539 seconds (9:59 min:sec).

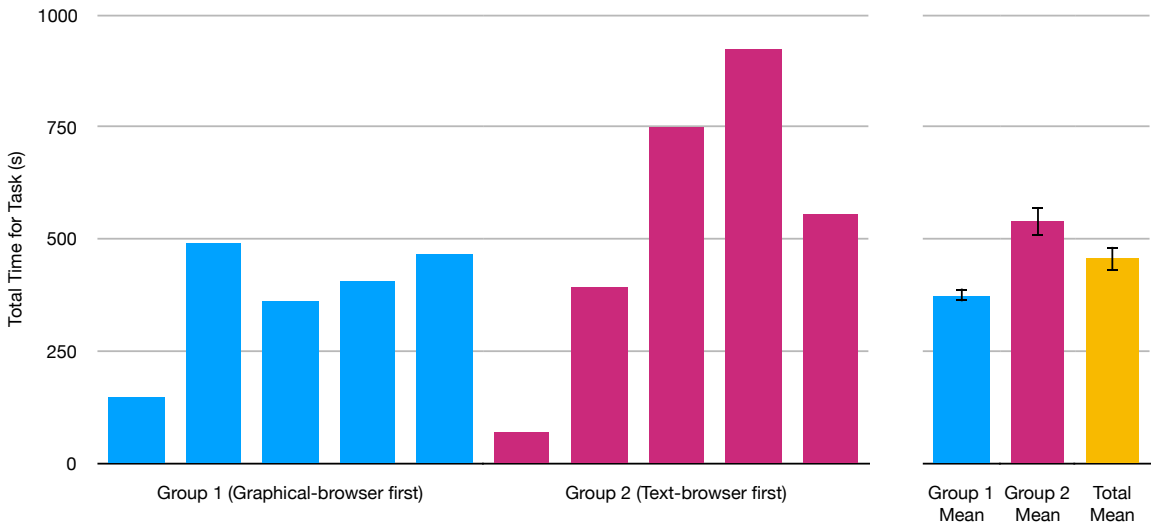

Figure 4.10: Total Time for User to Complete Task

4.3.2 Article Viewing Time

Figures 4.11 and 4.12 depict the amount of time participants spent viewing articles. Figure 4.11 details the time in seconds whereas Figure 4.12 presents the time as a proportion of the entire time the user spent on the task. The bottom bars of Figure 4.12 represent the

proportion of time spent viewing articles and the top bars delineate the time spent browsing the visualization. Similar to Figure 4.10, the data is separated by depending on the method with which the participant performed the search task first. To the right of the graph is a sub-graph illustrating the mean for each group and the total mean. On average, individuals in groups as well as the participants as a whole spent approximately half the time perusing the article text and the other half browsing the visualization.

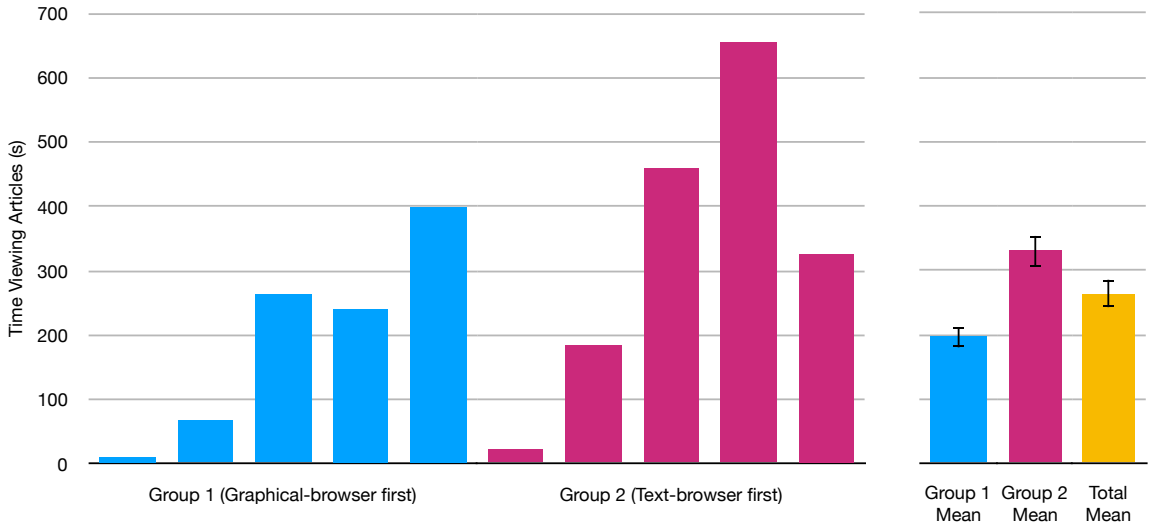

Figure 4.11: Time Participants Viewed Articles

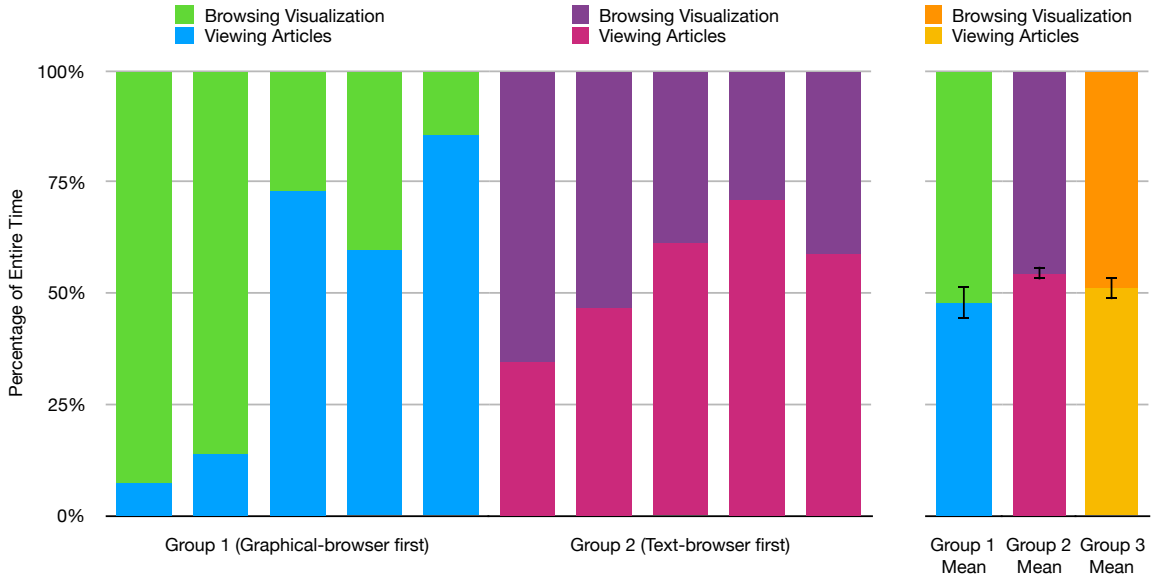

Figure 4.12: Percentage of Time Participant Viewed Article/Browser Versus Total Time

4.3.3 Count of Results Viewed

The last two tracking figures, Figure 4.13 and Figure 4.14, detail the number of bubbles viewed and the number of article links clicked, respectively. Figure 4.13 reveals the average number of bubbles all users viewed the contents of was 11, and on average participants in both groups explored 11 bubbles. In Figure 4.14, the total mean number of links clicked was 3.6, and Group 1 looked at 3 articles on average while users in Group 2 looked at 4.2 articles on average.

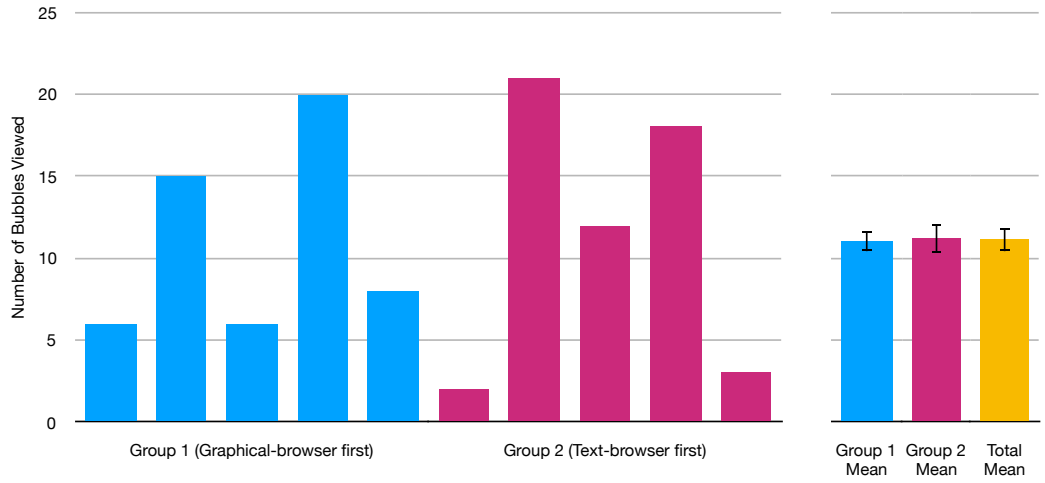

Figure 4.13: Number of Bubbles Participants Viewed

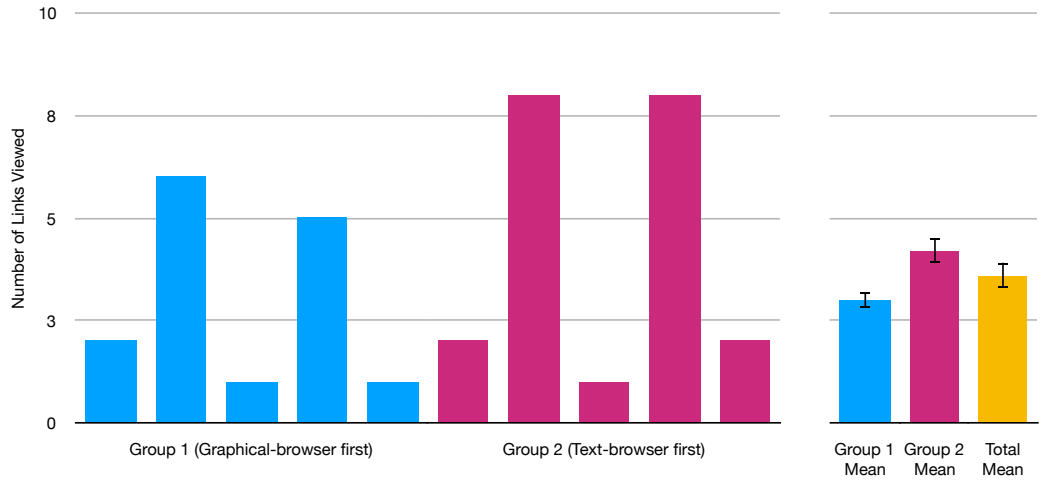

Figure 4.14: Number of Links Participants Viewed

Discussion

Overall, the visualization is designed to initially provide an overview of a large set of search results. Additionally, the interface provides an interactive method for browsing the results and narrowing the scope to find particular information within the results. To this end, similar articles are grouped together and positioned in bubbles which display word-clouds containing terms representative of the contents. The motivation behind this design is to provide the user with an initial summary of the search results, and after viewing the high-level visualization, the application affords the user the capability to narrow down the search parameters to find specific information.

To this end, various interface features provide various functionalities such as selecting a term from a bubble to re-execute the search, dragging links from one bubble into another to refine the search, and merging entire bubbles allowing the user to utilize the new sub-set of articles to further narrow the search. To analyze effectiveness of the application versus a standard text-based search bowser, a user study was performed with tracking data recorded and a survey given to generate the user feedback on the effectiveness of the visualization.

Presented is an assessment of the advantages and disadvantages of the visualization system and design and an analysis of the data collected during the user study. At the conclusion are discussed some thoughts on the future work which can be done to improve and evaluate the suitability of such a visualization for large search results.

5.1 Design Assessment

The system architecture follows a client-server approach where the visualization is rendered on the client and the search is executed on the server-side. The design of the visualization provides for the user to view and browse large-scale data, observe the overall structure of the data-set, and refine a search to find specific information within the results. To meet this end, the application provides numerous ways to interface with the visualization which allow the user to intuitively navigate the search results.

5.1.1 Overall System

In terms of the system as a whole, discussed in section 3.1, the visualization is implemented in JavaScript using libraries from D3. As such, the software runs in a web-browser, providing cross-platform flexibility and portability to the application. The visualization was developed and tested utilizing Firefox; however, minimal testing confirms the application runs in other browsers such as Chrome, Edge, and Safari. Additionally, the web-browser provides the added benefit of providing access to various input modes aside from a point-and-click interface since touch gestures are built into modern browsers. A main drawback of utilizing a browser involves the storage of search data for later analysis. Web-browsers typically disallow scripts from saving data to the client machine's disk. However, the user could configure web-storage to save search data, or the server script could be amended to persist data remotely.

The system architecture of the design, as described in section 3.2, provides for flexibility and scalability. Since the server handles the actual search and parsing of results, the client is free to render the visualization. The downside is the client must wait for the server to respond with the request, resulting in a delay between when the user performs an action requiring use of the search server. However, in the meantime the visualization will continue functioning with no additional processing required from the client machine.

In addition to the server allowing the client to run independently, the design allows for flexibility in terms of the search engine. To access the search, the client generates an HTTP request to the URL of the search script. If desired, different search scripts could be constructed to utilize different search engines such as Google or Bing. Additionally, the search script could use various metrics for parsing and grouping the results. For this implementation, K-Means clustering was used to group the results; however, various other methods could be utilized in order to provide better grouping of documents. For example, an LDA analysis could provide topics with which to group the documents. As long as the results are formatted to match the structure depicted in Table 3.5 the visualization is capable of displaying the results.

5.1.2 Search Visualization

The visualization design offers the user the capability of viewing the overall structure of a large set of search results. This means is achieved by visualizing the data in a tree-structure where the root represents the initial search and branches off each node are sub-sets of the search grouped by similarity. A sub-set of results is represented in the visualization as a word-cloud contained within a bubble.

Thus, the relatively large number of results from an initial search are broken down into smaller, more manageable sub-sets. The main advantage of this approach from a human-based perspective empowers the user with a method of seeing an overview of how the results are structured. After the user has browsed the visualization and added or removed bubbles, the structure of the tree visualization still represents an overview of the data. In this case, different branches will represent a further refinement in a specific direction, and each child node represents some transformation from its parent node.

As seen above, with each refinement of the search, a new node is created connected to its parent node. Accordingly, each node is allowed only one parent. This approach ensures the visualization will remain uncluttered from the extra connections required if

nodes were to have multiple parents. Additionally, the logic behind connecting new nodes to multiple parent nodes does not fit with what the visualization intends. In effect, the tree representation is a directed graph where each node represents a sub-set of results and the edges out of a node represent some transformation or refinement on the search query which resulted in the creation of the node. Therefore, giving a node multiple parents would imply multiple nodes were involved in the refinement of the search resulting in that node. It is possible to utilize multiple node data via the merge function; however there is no capability to separate previously merged nodes. If there were an un-merge function, it would make sense for a node to have multiple parents.

Certain subjects may have a relatively large number of associated results while other topics may have significantly fewer results. If desired, the user can further explore the larger group, or the smaller set can be expanded to provide a larger number of results. However, unless the user hovers over a bubble, there is no graphical representation of the relative number of results. An obvious method to represent the amount of results a node represents would be to scale the size of the circle to be proportional to the item count. However, for this design, it was decided to give each bubble the same static radius. Making the nodes uniform in size provides an agnostic approach to presenting the data. If the bubbles were variable in radius to represent the relative number of items contained, some users may take that to infer some nodes of the graph are more relevant to the search instead of merely containing more results. In fact, this representation is utilized within the word-clouds, as terms which have a stronger relation to the grouping of articles appear relatively larger in size than other terms.

5.1.3 Interface and Gestures

The design of the visualization allows for a high degree of interactivity with the application. As opposed to using keyboard short-cuts with hot-keys which need to be memorized, a user can quickly and naturally learn how to navigate the visualization. The simplicity

of the input interface allows the user to focus on browsing the visualization instead of remembering how certain functions are executed.

The benefit of utilizing the D3 library methods for generating a force-directed graph is the visualization is responsive to the user moving the nodes around and will keep the graph stable. If the user wishes to rearrange bubbles, they can be clicked or touched and dragged and D3 will continuously update the positions of all nodes. Once the graph is arranged to the user's requirements, the visualization can still be navigated by panning the canvas, which does not disturb the graph.

In the initial phases of implementation, the menu function selectors were not drawn around the bubble, yet they were buttons appearing at the bottom of the visualization. This approach was simpler and easier to implement; however, it was decided to move the menu functions to around the bubbles for a couple of reasons. First, when doing the select action, when the function buttons were located at the bottom of the screen, the user would be required to press the select button to enter selection mode and then press another button to leave, such as the add or move button. By placing the select button in the menu around the bubble, it is possible to simply press *Select* to select and un-select the particular bubble. Similarly, the delete button originally changed the mode the application was in, and touching or clicking a bubble would prompt the user if they wanted to delete the node. To leave the delete mode, the user had to again change to a different mode to continue.

5.2 Analysis of User Study Data

From the user study, questionnaire responses and the tracking data both serve as valuable metrics by which the visualization can be measured. The data presented in 4.2 and 4.3 are discussed in the following two sections. Additionally, the open-ended question in the survey provides beneficial feedback on the application.

5.2.1 Questionnaire

When compared to a standard text-based manner of viewing results, participants felt the graphical search browser performed at least on par or better. As seen in Figure 4.2 , participants describe liking the graphical browser at least as much as the text-browser. In Figure 4.4, participants felt the graphical interface was at least as convenient as a text-based mode and none felt it was less convenient. Figure 4.3 reveals that all but one respondent felt the graphical search allowed them to complete the task quicker than the text browser. Thus, the visualization, for the most part, was viewed on the same level with typical search browsing methods.

In terms of the visualization's capability to present a synopsis of the search results, Figures 4.3 and 4.5 again demonstrate the visualization performs on par with standard browsing methods. While it was hoped that the visualization would perform much better than a text-based browser, the results are promising in that the graphical browser was at least slightly preferred in terms of viewing large data sets.

When asked about the various interaction features, results were varied. Figure 4.7 found that two-thirds of the participants utilized the touch screen over the mouse, and of those users, most found the touchscreen made browsing easier, as seen in Figure 4.8. While use of a touch-interface is not novel to this application, it is positive to find the touchscreen functions as a natural interface to interacting with the visualization. When asked about the menus, Figure 4.9 illustrates mixed opinions.

While most respondents felt the menus were moderately effective, an equal amount viewed them as either slightly effective or not at all. This result may be due to the fact that few of the participants needed to utilize the menu in order to complete the task. The most useful function with regard to the tasks performed would be *add*, with none needing to *select*, *merge*, nor *delete* any nodes. Thus, the data could be interpreted to signify the functions are mostly unnecessary, or perhaps they would be more useful when dealing with larger data-sets, as the number of results seen during the study was set to 50.

The final part of the user survey to examine, the open-ended question, provides some observations about the application, some of which section 5.4 also references. From the responses listed in Table 4.3, in terms of the interactivity of the visualization, the most notable takeaway is the need to put more work in to the interactivity. Since the application falls more into the category of proof-of-concept or prototype, these statements are understandable. The next category in Table 4.3 includes responses dealing with the visualization. These responses gave specific suggestions such as making the bubble size correspond to the article count and being able to utilize specific search terms. One response indicated providing more training time for the user, which could be achieved by carrying out the study by having the user perform multiple tasks in the visualization instead of just one. Another comment, which may relate to the issue of finding information within an article, suggests formatting the text of the article. Since the articles were shown as a plain text file, it might be understandable that participants spend roughly half their time reading through the articles; formatting the text to be more visually agreeable may decrease this proportion. A final comment regards the neural network provided which executes the search. At the time of the user studies, the neural network utilized had been trained on a limited data-set, and further work is being done to improve the quality of the results retrieved via this method.

5.2.2 Task Time

In relation to the time needed to complete the task gathered from the tracking data, Table 4.5 shows users performed the tasks about half the time on average with the standard text browser compared to the graphical browser. With a p-value of 0.016 from the t-test, this result appears to be statistically significant. There may be several factors contributing to this difference. As stated in the section Task Time, the text browser was only capable of displaying the first 10 results from the search, whereas the graphical browser utilized the top 50 results. With more results to navigate through, it is understandable the participants would need more time to complete the task. Additionally, each participant was given a short

overview of the visualization and its various functions and allowed to figure out the best mode for navigating the results. Thus, the disparity between task times could be attributed to acclimating to a new system for browsing. The participants only performed one task, and for further studies it would be beneficial to have the user perform more tasks. After several tasks, the user should have better understanding of how to utilize the visualization, allowing for a better quantitative analysis.

While the study shows the task can be completed quicker in the text browser, other statistics based on the difference between the two groups show having prior knowledge of the search doesn't necessarily equate in a quicker task time. It was discussed in section 4.3 that the collected data summarized in Table 4.4 does not show a statistically significant difference between the groups. The data in Figure 4.10 does show a trend that users with no prior knowledge of the search results were able to complete the task in less time, yet since the study size was not large enough, no strong conclusion can be made to this result. Additionally, participants in Group 1 spent less time viewing the articles than those in Group 2. These results are interesting, yet they go slightly against what might be expected. Since the participants in Group 2 had already found an article containing the desired information, it would be assumed that they would simply find the same article in the graphical browser. However, the participants in Group 2 did not always look for the article they had already viewed. In fact, it was observed that some participants avoided looking at the same article, although this action was not disallowed. In future studies, it will be necessary to make all the restricted and allowed actions of the participant well understood.

In Figure 4.12, the percentage of time the user spent reading articles is shown with respect to the total time it took to complete the task. Participants from both Groups 1 and 2 spent nearly half the time browsing the visualization and the other half viewing the articles. This statistic reveals that regardless of the amount of time the participant took to find the requested information, that time was typically evenly split between browsing the visualization for articles related to the requested information and reading through articles

for the specific item. Because of this fact, a few details can be surmised. As with the discussion above relating to total time required to complete the task, this statistic may deviate from what one might naturally conclude. Since individuals in Group 2 had already viewed an article with the specified information, when examining articles with the visualization it might be expected they would skip looking at articles that were not the one they had already viewed. Upon discovering that article, it might also be assumed the participant would skim to the location of the requested information. It does not appear participants in Group 2 made that decision; however, from Figure 4.12 they performed the task with the same proficiency as Group 1. This fact could also indicate users in both groups needed similar amounts of time to acclimate to the visualization before focusing on the task. One possible way to measure this factor would be to have participants perform multiple tasks and measure the change in time required to complete each task.

Related to the time required to complete the task, Figures 4.13 and 4.14 depict the number of bubbles participant viewed the contents of and the number of articles they clicked to read the contents. Again, section 4.3 states there is no statistical significance in the difference between the groups. Figure 4.13 reflects the total number of bubbles viewed including duplicates. It may have been the case where an individual would scan the contents of one bubble and move on to a different node, later coming back to the first bubble they explored. The figures indicate participants in Groups 1 and 2 perused the same number of bubbles. This fact indicates that regardless of having knowledge of what specific article contained the desired information, all participants were required to browse the visualization an equal amount. Figure 4.14 displays participants in Group 2 looked at slightly more articles on average than Group 1. As the above discussion of individuals in Group 2 spending more time on the task, this statistic again reveals that those users did not merely look for the article they had viewed in the text-browser. Whether they were avoiding the article or not is unknown. It could also be the case they could not find the exact article; many articles in the data set were very similar with small addendum and revisions to the

same article. It is possible participants in Group 2 attempted to view the same article they had previously encountered, yet although the headline was the same the contents were not.

5.3 Implementation Experience

The implementation of the application introduced various challenges. First, before deciding on a web-based application utilizing D3, other graphics frameworks were investigated. In [37], several software packages for visualizing large graphs are mentioned including Gephi (Java), Graphviz (C, C++, Python) , Open Graph Drawing Framework and Tulip (C++). In addition, the Visualization Toolkit (VTK) provides functionality for 2-D and 3-D graphics as well as algorithms for generating graphical representations for data structures such as graphs and trees. While these platforms are powerful visualization tools, D3 was chosen partly for its portability and high degree of interactivity available.

After deciding on D3, the challenge arose of learning JavaScript and HTML. Fortunately, D3 provides many on-line examples on how to generate force graphs and word-clouds, and several code-snippets regarding handling interaction events. Still, some parts of the application do not work entirely as expected. For instance, the event that is triggered when the mouse enters and leaves the menu is not entirely reliable, and the callback function does not execute. Because of this situation, the menu may not disappear as intended when the mouse exits.

Another challenge with JavaScript and HTML regards the ability to drag links from the results list into a bubble. When utilizing the mouse, the browser handles the drag event as expected, moving the text of the link with the mouse cursor. When the mouse button is released, an event is triggered which places the link in the corresponding bubble. However, when dragging a link with the touch interface, the browser does not automatically handle the event. Instead, a separate function was required to handle the touch-drag and touch-drop actions. Thus, the mouse and touch drag-and-drop for links appear different in the

application.

A final challenge of the implementation was generating a data-set to be visualized. While the application can operate with dummy data, to evaluate the performance of the visualization actual search results were required. To this end, various search engines and document clustering methods were investigated. Both Google and Bing each provide an Application Programming Interface (API) to access web search results; however, unless one were to pay for the service, the APIs limit an account to roughly a dozen results per day. While useful for generating initial data to build and test the visualization, this limitation disallows access to large data-sets which are the focus of the project.

Another search engine, YaCy [38] was also investigated. YaCy is a peer-to-peer search engine where each peer shares indices of sites it has crawled with other peers in the network. YaCy is free to use, and at first the results appeared promising. After working with YaCy, it was discovered that the results were inconsistent, and many links returned in a given search appeared unrelated to the query. Additionally, performing a search with the same terms did not always return the same results. Executing a search one minute may return hundreds of results, while the same search a few seconds later would return less than ten.

Finally, once a data-set of search results was obtained, the documents needed to be groups and parsed for the visualization. Various approaches were considered such as faceted search [39] and mining database structures [40]. Since the aim of this project was not to develop new semantic understanding methods, it was settled on using simple K-Means clustering for document grouping and TFIDF and NMF for topic term generation, since these methods are dependable data mining methods.

5.4 Future Work

As work on visualizing large sets of search results continues, more analysis would be needed on how to task and measure an an open-ended browsing experience. For this project, the user study focused on finding specific items within the search results. It is difficult and likely costly to devise an experiment where the participants are given an ambiguous goal or no goal at all. With the time and finances available to this project, undertaking such a study would be infeasible. Hopefully the success of initial studies can show the effectiveness of graphical search methods to further research into the area.

Since the visualization is focused on large data-sets, modifications to the tree representation may help facilitate browsing. In the open-ended responses from the user survey questionnaire 4.2.4, one response mentioned the size of the bubble could be representative of the number of articles contained. However, another approach to dynamically sizing the nodes could be implementation of a hyperbolic tree visualization [41], an example of which is shown in Figure 5.1. In this case, the node currently being browsed would appear the largest, and as the distance from the focused node increases, the size of the bubbles would logarithmically decrease and practically disappear far enough out. This approach may facilitate the user to focus on results most similar to the ones they are interested in with the nodes not as related vanishing at the edges, yet still browse-able.

There is no restriction disallowing multiple nodes from containing the same entry; however, similar to the above discussion in section 5.1.2 about a node having multiple parents, there is currently no graphical depiction of how similar two different bubbles are. One possibility would be to have similar nodes gravitate toward one another. Additionally, bubbles could have color arcs at the edge of the circle where the color represents a different node and the size of the arc is relative to the similarity between articles. Figure 5.2 depicts such a concept wherein each cluster is represented by a color. In the Figure 5.2 mock-up, cluster 1 and 2 are relatively similar, so the color arcs for the corresponding group is bigger than the color arc they have for cluster 3. This is similar to the approach described in [4],

and in fact they take the further step of rendering the background canvas in the color of the particular topic cluster. This approach is also similar to that utilized in VOWL [24], where different classes of objects are color-coded. The main restriction on this approach is the color palette chosen, as the colors for groups would need to be distinct enough for the user to distinguish the groups. This limitation would also reduce the number of clusters available to be visualized, as it would be confusing to have two groups with only slightly different shades of the same color.

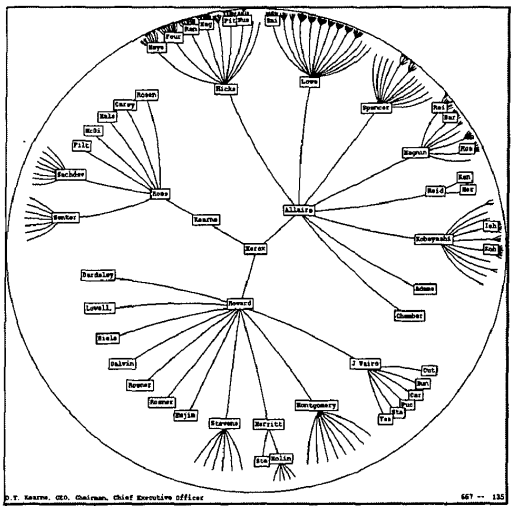

Figure 5.1: Hyper-tree Example

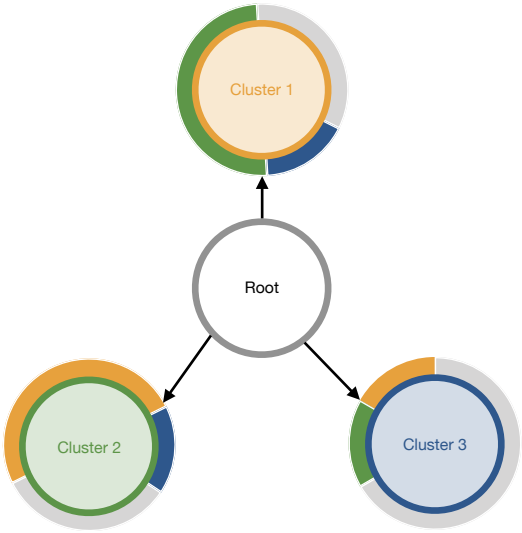

Figure 5.2: Possible Method for Visualizing Bubble Similarity

Another aspect of the visualization that could be investigated is the possibility of utilizing media other than documents, such as images and video. For instance, images could be clustered based on feature similarities within the images or terms associated with the image. Hovering over a bubble could expand the images to appear around the bubble, similar to how the menu appears. With respect to videos, the user could drag the mouse or finger around the outside of the bubble to scrub through the video.

Other features which could be integrated into future iterations include more flexibility when merging nodes. The user may wish to perform a union of the contents, which is how the application functions currently; however they may also desire either only similar or

dissimilar items. From the interaction standpoint, instead of adding new menu options for ”Union” and ”Intersection”, the menu could be modified to react to a ”slide” gesture where a slide in one direction could signify union and the other direction, intersection. Additionally, touch gestures could replace many of the menu functions. For instance, the user could quickly swipe a node they wished to delete toward the edge of the screen. Dragging two nodes close to each other could initiate a merge action, and pinching out on a node which had been merged could un-merge the bubble.

Lastly, while the system was developed and tested on desktop machines, mobile devices such as tablets and smart-phones are a natural platform for this type of application. While some of the features have been tested and work on mobile devices, more work would need to be done to port the application to those platforms. Additional platforms the visualization might benefit from are virtual and augmented reality devices. In such a situation, the visualization could be expanded to a 3-dimensional graph to represent more of the relationships between results.

5.5 Conclusion

Presented with a large amount of search results, users may have difficulty making sense of the information and patterns hidden within. The visualization designed and implemented for this project concerns interactively browsing large document sets from a search. To meet this end, the set of results is displayed graphically as a tree, and the nodes of the tree are similar documents shown in a bubble with a word-cloud of terms relevant to the results contained. Users can interact with the visualization by dragging nodes around to rearrange the structure, refine the search by selecting terms or articles within a particular bubble, and perform other actions such as merging and deleting nodes. The visualization was evaluated with a user study wherein users were given a specific data item to find within the visualization. The statistics from the evaluation do not show strong confidence in the

result; nonetheless, the data trends toward the fact that the visualization performs as well or better than a standard text-based browser. Future work on the application would involve advanced user studies to understand the effectiveness vis a vis an unguided search task of large search results.

Bibliography

[1] A. A. B., de Oliveira Maria Cristina F., and P. F. V., "Seeing beyond reading: a survey on visual text analytics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 6, pp. 476–492, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1071

[2] S. H., S. M., S. A., K. D., and D. O., "Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives," *Computer Graphics Forum*, vol. 31, no. 3pt3, pp. 1135–1144, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03106.x

[3] Google, "Google ngram viewer." [Online]. Available: https://books.google.com/ngrams

[4] O. D., S. H., R. C., G. I., and D. O., "Comparative exploration of document collections: a visual analytics approach," *Computer Graphics Forum*, vol. 33, no. 3, pp. 201–210, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12376

[5] L. Hanseung, K. Jaeyeon, C. Jaegul, S. John, and P. Haesun, "ivisclustering: An interactive visual document clustering via topic modeling," *Computer*

Graphics Forum, vol. 31, no. 3pt3, pp. 1155–1164, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03108.x

[6] S. Lohmann, V. Link, E. Marbach, and S. Negru, "Webvowl: Web-based visualization of ontologies," in *Knowledge Engineering and Knowledge Management*, P. Lambrix, E. Hyvönen, E. Blomqvist, V. Presutti, G. Qi, U. Sattler, Y. Ding, and C. Ghidini, Eds. Cham: Springer International Publishing, 2015, pp. 154–158.

[7] "Graph visualization and social network analysis software — navigator - touchgraph.com," May 2018. [Online]. Available: http://www.touchgraph.com/navigator

[8] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107 – 117, 1998, proceedings of the Seventh International World Wide Web Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016975529800110X

[9] T. Ruppert, M. Staab, A. Bannach, H. Lücke-Tieke, J. Bernard, A. Kuijper, and J. Kohlhammer, "Visual interactive creation and validation of text clustering workflows to explore document collections," *Electronic Imaging*, vol. 2017, no. 1, pp. 46–57, 2017. [Online]. Available: https://www.ingentaconnect.com/content/ist/ei/2017/00002017/00000001/art00006

[10] W. Dou, X. Wang, R. Chang, and W. Ribarsky, "Paralleltopics: A probabilistic approach to exploring document collections," in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, Oct 2011, pp. 231–240.

[11] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Data Mining*, 2nd ed. Cambridge University Press, 2014, pp. 1–18.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation David," *Journal of Machine Learning Research2*, 2003.

[13] D. Sculley, "Web-scale k-means clustering," *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010. [Online]. Available: http://dx.doi.org/10.1145/1772690.1772862

[14] J. Harris, "Word clouds considered harmful." [Online]. Available: http://www.niemanlab.org/2011/10/word-clouds-considered-harmful/

[15] J. Seo and B. Shneiderman, "A rank-by-feature framework for interactive exploration of multidimensional data," *Information Visualization*, vol. 4, no. 2, pp. 96–113, May 2005. [Online]. Available: http://dx.doi.org/10.1057/palgrave.ivs.9500091

[16] R. Kosara, F. Bendix, and H. Hauser, "Parallel sets: interactive exploration and visual analysis of categorical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 558–568, July 2006.

[17] S. L'Yi, B. Ko, D. Shin, Y.-J. Cho, J. Lee, B. Kim, and J. Seo, "Xclusim: a visual analytics tool for interactively comparing multiple clustering results of bioinformatics data," *BMC Bioinformatics*, vol. 16, no. 11, p. S5, Aug 2015.

[18] B. Alsallakh, W. Aigner, S. Miksch, and H. Hauser, "Radial sets: Interactive visual analysis of large overlapping sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2496–2505, Dec 2013.

[19] S. Liu, M. X. Zhou, S. Pan, Y. Song, W. Qian, W. Cai, and X. Lian, "Tiara: Interactive, topic-based visual text summarization and analysis," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 2, pp. 25:1–25:28, Feb. 2012. [Online]. Available: http://doi.acm.org/10.1145/2089094.2089101

[20] T. May, A. Bannach, J. Davey, T. Ruppert, and J. Kohlhammer, "Guiding feature subset selection with an interactive visualization," in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, Oct 2011, pp. 111–120.

[21] J. Chuang, C. D. Manning, and J. Heer, "Termite: Visualization techniques for assessing textual topic models," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ser. AVI '12. New York, NY, USA: ACM, 2012, pp. 74–77. [Online]. Available: http://doi.acm.org/10.1145/2254556.2254572

[22] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow, "Visualizing the non-visual: spatial analysis and interaction with information from text documents," in *Proceedings of Visualization 1995 Conference*, Oct 1995, pp. 51–58.

[23] M. Bostock, "D3.js - data-driven documents," 2017. [Online]. Available: https://d3js.org

[24] S. Lohmann, S. Negru, F. Haag, and T. Ertl, "Vowl 2: User-oriented visualization of ontologies," in *Knowledge Engineering and Knowledge Management*, K. Janowicz, S. Schlobach, P. Lambrix, and E. Hyvönen, Eds. Cham: Springer International Publishing, 2014, pp. 266–281.

[25] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969 – 978, 2009, special Section on Graphical Models and Information Retrieval. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888613X08001813

[26] Mozilla. (2018) Firefox. [Online]. Available: https://www.mozilla.org/en-US/firefox/60.0.2/releasenotes/

[27] N. Cahill, "Split.js," 2018. [Online]. Available: https://split.js.org

[28] M. Bostock, "d3-3.x-api-reference/force-layout." [Online]. Available: https://github.com/d3/d3-3.x-api-reference/blob/master/Force-Layout.md

[29] J. Davies, "Word cloud generator." [Online]. Available: https://www.jasondavies.com/wordcloud/about/

[30] J. Steele and N. Iliinsky, *Beautiful Visualization: Looking at Data through the Eyes of Experts*, ser. Theory in practice series. O'Reilly Media, 2010. [Online]. Available: https://books.google.com/books?id=TKh6fdlKwfMC

[31] C. C. Aggarwal and C. Zhai, *A Survey of Text Clustering Algorithms*. Boston, MA: Springer US, 2012, pp. 77–128. [Online]. Available: https://doi.org/10.1007/978-1-4614-3223-4_4

[32] M. Cordeil, T. Dwyer, K. Klein, B. Laha, K. Marriott, and B. H. Thomas, "Immersive collaborative analysis of network connectivity: Cave-style or head-mounted display?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 441–450, Jan 2017.

[33] M. Feng, C. Deng, E. M. Peck, and L. Harrison, "The effects of adding search functionality to interactive visualizations on the web," *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, 2018. [Online]. Available: http://dx.doi.org/10.1145/3173574.3173711

[34] T. Ruppert, A. Bannach, J. Bernard, H. Lücke-Tieke, A. Ulmer, and J. Kohlhammer, "Supporting collaborative political decision making," *Proceedings of the 9th International Symposium on Visual Information Communication and Interaction - VINCI '16*, 2016. [Online]. Available: http://dx.doi.org/10.1145/2968220.2968223

[35] C. Forsell, "A guide to scientific evaluation in information visualization," in *2010 14th International Conference Information Visualisation(IV)*, vol. 00, July 2010, pp. 162–169. [Online]. Available: doi.ieeecomputersociety.org/10.1109/IV.2010.33

[36] S. Carpendale, *Evaluating Information Visualizations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 19–45. [Online]. Available: https://doi.org/10.1007/978-3-540-70956-5_2

[37] H. Yifan and S. Lei, "Visualizing large graphs," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 2, pp. 115–136, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1343

[38] YaCy, "Yacy," 2018. [Online]. Available: www.yacy.net

[39] D. Tunkelang, "Faceted search," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 1, no. 1, pp. 1–80, Jan 2009. [Online]. Available: http://dx.doi.org/10.2200/S00190ED1V01Y200904ICR005

[40] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk, "Mining database structure; or, how to build a data quality browser," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '02. New York, NY, USA: ACM, 2002, pp. 240–251. [Online]. Available: http://doi.acm.org/10.1145/564691.564719

[41] J. Lamping, R. Rao, and P. Pirolli, "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies," *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*, 1995. [Online]. Available: http://dx.doi.org/10.1145/223904.223956