Visualization of Search Results of Large Document Sets

James D Anderson and Thomas Wischgoll; Wright State University; Dayton, OH

Abstract

When presented with many search results, finding information or patterns within the data poses a challenge. This paper 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 demonstrated that in comparison to a standard text-browser for viewing search results, the visualization performs commensurate or better on most metrics.

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. Furthermore, the most applications currently available for browsing results are purely text-based which display a listing of results which may or may not be ranked. If the results are ranked, there may be documents hidden deep in the list the analyst may wish to view; however, since the results are lower in the list, those results have less likelihood of being seen 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 as 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 presents a design, shown in figure 1 which visualizes a 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 visualization evaluation was conducted with participants performing a search task and afterward providing feedback on the application.

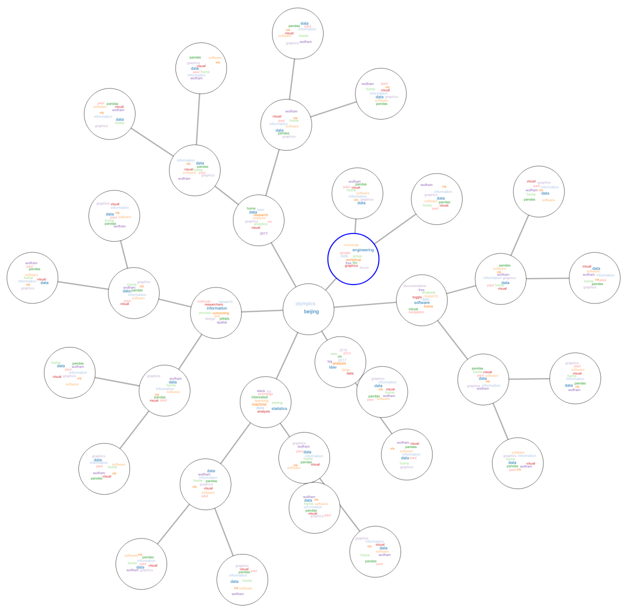


Figure 1. *Search Graph Visualization*

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, because the structure of many data sets can be visualized as a graph, there has been much work on visualizing these large relational data structures.

Traditional Search

Traditionally, search results of text-based data are viewed in a list format. For example, the Google search browser provides a summary of the topic in addition to the results listed as text,. Typically, the results are shown sorted by a ranking metric such as Google's uses a PageRank [1]. For Google results, the anchor text, or 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.

Document Set Visualizations

To visualize textual information, various methods have been developed to reduce the complexity from hundreds and thousands of terms to a more sizable and human-understandable space. Two common methods, document topic generation and clustering [2, 3], use term frequency-inverse document frequency (TFIDF) with word-vectors and Latent Dirichlet allocation (LDA). LDA is a statistical model which assigns probabilities to topics based on the collection of terms within a document [4]. With LDA, connections between documents can be made even though the documents themselves may consist of mostly disjoint sets of terms.

Once terms and topics are generated from the documents, a common visualization method is a word-cloud. With this technique, the terms are presented with their font sizes proportional to the significance or probability of being related to the data set and typically arranged as to minimize empty space,. Rolled-out Wordles [5] demonstrates a heuristic for building word clouds by removing overlaps between elements. TagCrowd represents documents as word-clouds with the size of a word being proportional to its frequency within the text [6].

Another technique called Hierarchical point placement (HiPP) [6] has circles, or "bubbles" with proximities proportional to similarity between the document sets, while the circles represent similar documents. DiTop-View [7], a visualization method with bubbles and word-clouds, partitions the canvas into different background colors which represent major topic areas.

Many visualization methods utilize document clustering to group semantically similar documents. One such, iVisClustering [8], clusters documents by topic utilizing LDA to generate a graph visualization where closely related documents are grouped together with a display of topic words.

Graph Visualizations

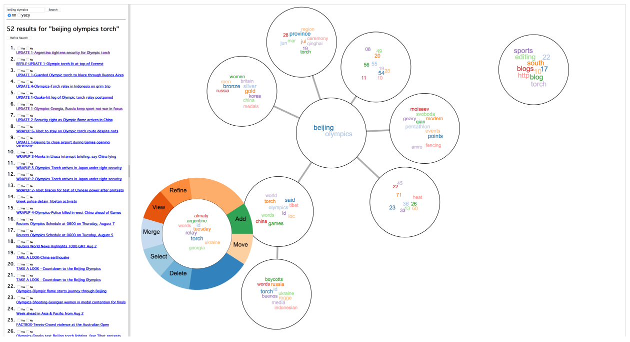
Several graphical frameworks for depicting relational data have been developed, and this paper's design takes inspiration from ontology-related visualizations. In particular, WebVOWL [9] is a web-based visualization tool for graphic display of an ontology which utilizes Scalable Vector Graphics (SVG) and Cascading Style Sheets (CSS) along with the JavaScript library Data Driven Documents (D3) [10] to display force-directed graphs. This approach allows for dynamic addition, removal, and repositioning of nodes, as the visualization will adjust to the change in graph structure.

Another inspiration for this project is the TouchGraph Navigator [11]. 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.

In contrast to all the described applications, 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 documents more associated with the desired term. Furthermore, the user can select articles to receive additional results similar to the chosen documents.

Implementation

The search visualization 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.



beijing olympics | Search
● nn ○ yacy

52 results for "beijing olympics torch"

Refine Search

1. ☐ Yes ☐ No UPDATE 1-Argentina tightens security for Olympic torch
2. ☐ Yes ☐ No REFILE-UPDATE 1-Olympic torch lit at top of Everest
3. ☐ Yes ☐ No UPDATE 1-Guarded Olympic torch to blaze through Buenos Aires
4. ☐ Yes ☐ No UPDATE 4-Olympics-Torch relay in Indonesia on grim trip
5. ☐ Yes ☐ No UPDATE 1-Quake-hit leg of Olympic torch relay postponed
6. ☐ Yes ☐ No UPDATE 1-Olympics-Georgia, Russia keep sport not war in focus
7. ☐ Yes ☐ No UPDATE 2-Security tight as Olympic flame arrives in China
8. ☐ Yes ☐ No WRAPUP 6-Tibet to stay on Olympic torch route despite riots
9. ☐ Yes ☐ No UPDATE 1-Beijing to close airport during Games opening ceremony
10. ☐ Yes ☐ No WRAPUP 3-Monks in Lhasa interrupt briefing, say China lying
11. ☐ Yes ☐ No WRAPUP 3-Olympics-Torch arrives in Japan under tight security
12. ☐ Yes ☐ No WRAPUP 2-Olympics-Torch arrives in Japan under tight security
13. ☐ Yes ☐ No WRAPUP 2-Tibet braces for test of Chinese power after protests
14. ☐ Yes ☐ No Greek police detain Tibetan activists
15. ☐ Yes ☐ No WRAPUP 4-Olympics-Police killed in west China ahead of Games
16. ☐ Yes ☐ No Reuters Olympics Schedule at 0600 on Thursday, August 7
17. ☐ Yes ☐ No Reuters Olympics Schedule at 0600 on Tuesday, August 5
18. ☐ Yes ☐ No Reuters World News Highlights 1000 GMT Aug 2
19. ☐ Yes ☐ No TAKE A LOOK-China earthquake
20. ☐ Yes ☐ No TAKE A LOOK - Countdown to the Beijing Olympics
21. ☐ Yes ☐ No TAKE A LOOK - Countdown to the Beijing Olympics
22. ☐ Yes ☐ No Olympics-Olympic flame starts journey through Beijing
23. ☐ Yes ☐ No Olympics-Shooting-Georgian women in medal contention for finals
24. ☐ Yes ☐ No Week ahead in Asia & Pacific from Aug 2
25. ☐ Yes ☐ No FACTBOX-Tennis-Crowd violence at the Australian Open
26. ☐ Yes ☐ No Olympics-Greeks test Beijing torch lighting, fear Tibet protests

Figure 2. *Visualization Screen Image*

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 [12]. For this visualization, a dataset of Reuters articles serves as the basis for search queries.

Client Visualization

As seen in Figure 2, the visualization is divided into three sections: input (top-left), output (bottom-left), and the graphical tree (right). In the input area, a standard text-input box allows the user to type in the initial search query. The text-based output of the search results appear as hyper-links, enabling the user to easily access the document. "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.

The right section of Figure 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. As the user clicks on documents, this bubble is populated with terms relating to that document. When the user initiates a search, a bubble will appear containing those search terms as the root of the new query. New bubbles are generated connected to the root populated with terms related to a sub-group of the entire search.

Main Search

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; 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 [13]. 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.

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. 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.

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. 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.

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.

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.

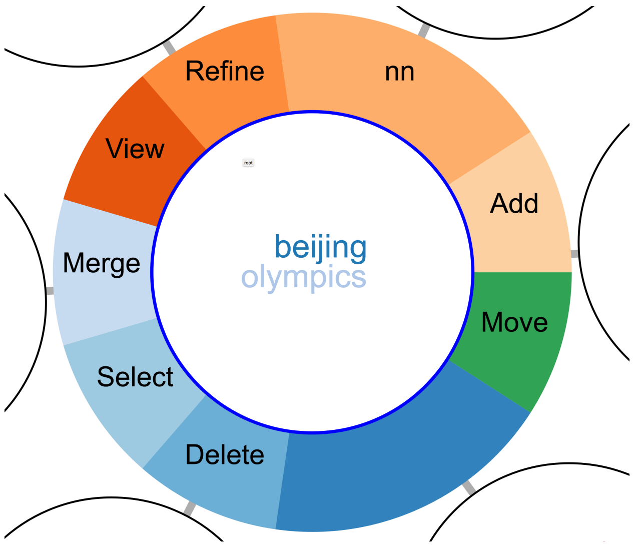


Figure 3. *Menu Detail*

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.

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. The user can drag-and-drop the terms in the *suggested search term bubble* into any of the search bubbles, allowing a new term to be utilized in a search refinement.

Additionally, the user can drag a search result link from the left-panel into one of the search bubbles. This action will cause the link to be checkmarked "Yes" to be utilized when refining results, as well as populating the target bubble with words relevant to the dragged link.

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 in figure 3 provides access to various functions which can be performed. 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* toggle the mode of interaction. *Move* mode, allows bubbles to freely move and the disables clicking on terms. In *Add* mode, terms become click-enabled, which causes the application to perform an additional search using that term and adding that result as a child of the current bubble.

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. For the SVG arc, the inner and outer radius can be specified, which 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.

Server-side Search

When the user enters a search query or chooses to refine the search parameters, an HTTP request is sent to a server hosing a search script which utilizes 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.

To execute the search and parse the results for visualization, a Python script is hosted on an Apache web server with Common Gateway Interface (CGI) enabled. 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) [14] for the document set. NMF is capable of extracting topics from a document set, and these topics can be utilized by the visualization.

Additionally, the search process utilizes semantic hashing [12], a machine learning approach to searching which involves training a neural network with the inputs being documents represented as word-vectors where at each feature is the frequency of a particular term. Semantic hashing also offers the capability to generate results similar to a given set of specific documents, providing the ability to refine search results. Bit-vectors of specific documents can be compared with vectors of other documents to find more semantically similar articles. The visualization incorporates this capability in two ways: 1) checking the "Yes" box next to a result and 2) dragging a document into a bubble. After performing either of these actions, the user may "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

To evaluate the application, a group of volunteer participants ($N = 12$) performed a task utilizing the graphical visualization search browser and afterward completed a user survey. Each participant broweed results from the same search query of "beijing olympics" in order to find a distinct piece of information. 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.

The surveys were designed to test the effectiveness and convenience of the visualization versus a normal text-based search browser. As such, the participant conducted the same task twice: once with the graphical-browser and once with the text-browser. 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.

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.

User Surveys

After completing the search tasks, participants completed a survey comprised of multiple choice questions and one open-ended question. A summary of the questionnaire statistical analysis is shown in Table 1. The listing shows the means of responses from Groups 1 and 2, as well as the overall mean, normalized between 1.0 and 0.0 with 1.0 corresponding to most favorable to the visualization and 0.0 least favorable. Also shown are the standard deviations (σ) for all responses to the particular question, which range from 0.15 to 0.31. A t-test was done for each question comparing 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.

Open-ended Responses

One open-ended question was asked in the survey: *In what way would you improve the search?*. Feedback statements generally relate to 3 different categories: Interactivity, the Visualization, and the Search Engine. 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 contained primarily numbers, and the other recommended more training of the neural network.

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

Summary of Survey Statistics

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

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.

Acknowledgments

The authors wish to thank Eric Nichols, Brad Minnery, and Michael Raymer. This work was supported in part by a grant from the Ohio Federal Research Network (ORFN).

References

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[2] Tobias Ruppert, Michael Staab, Andreas Bannach, Hendrik Lücke-Tieke, Jürgen Bernard, Arjan Kuijper, and Jörn Kohlhammer. Visual interactive creation and validation of text clustering workflows to explore document collections. *Electronic Imaging*, 2017(1):46–57, 2017.

[3] 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)*, pages 231–240, Oct 2011.

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation David. *Journal of Machine Learning Research2*, 2003.

[5] Strobelt H., Spicker M., Stoffel A., Keim D., and Deussen O. Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. *Computer Graphics Forum*, 31(3pt3):1135–1144, 2012.

[6] Alencar Aretha B., de Oliveira Maria Cristina F., and Paulovich Fernando V. Seeing beyond reading: a survey on visual text analytics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):476–492, 2012.

[7] Oelke D., Strobelt H., Rohrdantz C., Gurevych I., and Deussen O. Comparative exploration of document collections: a visual analytics approach. *Computer Graphics Forum*, 33(3):201–210, 2014.

[8] Lee Hanseung, Kihm Jaeyeon, Choo Jaegul, Stasko John, and Park Haesun. ivisclustering: An interactive visual document clustering via topic modeling. *Computer Graphics Forum*, 31(3pt3):1155–1164, 2012.

[9] Steffen Lohmann, Vincent Link, Eduard Marbach, and Stefan Negru. Webvowl: Web-based visualization of ontologies. In Patrick Lambrix, Eero Hyvönen, Eva Blomqvist, Valentina Presutti, Guilin Qi, Uli Sattler, Ying Ding, and Chiara Ghidini, editors, *Knowledge Engineering and Knowledge Management*, pages 154–158, Cham, 2015. Springer International Publishing.

[10] Mike Bostock. D3.js - data-driven documents, 2017.

[11] Graph visualization and social network analysis software — navigator - touchgraph.com, May 2018.

[12] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969 – 978, 2009. Special Section on Graphical Models and Information Retrieval.

[13] Mike Bostock. d3-3.x-api-reference/force-layout.

[14] Charu C. Aggarwal and ChengXiang Zhai. *A Survey of Text Clustering Algorithms*, pages 77–128. Springer US, Boston, MA, 2012.

Author Biography

James Anderson received his MS in computer engineering in 2018 from Wright State University, and is working on his PhD at the same institution. His research work is in collaboration with researchers at the Air Force Institute of Technology (AFIT) on simulating and analyzing flights involving automated aerial refueling. He has presented and published in the International Symposium on Visual Computing conference proceedings.

Thomas Wischgoll received his Master's degree in computer science in 1998 from the University of Kaiserslautern, Germany, and his PhD from the same institution in 2002. He was working as a post-doctoral researcher at the University of California, Irvine until 2005 and is currently an associate professor and the Director of Visualization Research at Wright State University. His research interests include large-scale visualization, flow and scientific visualization, as well as biomedical imaging and visualization. His research work in the field of large-scale, scientific visualization and analysis resulted in more than thirty peer-reviewed publications, including IEEE and ACM. Dr. Wischgoll is a member of ACM SIGGRAPH, IEEE Visualization & Graphics Technical Committee, and the IEEE Compute Society.