CubanSea: Cluster-Based Visualization of Search Results

Matthias Tilsner
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
matthias.tilsner@mun.ca

Abstract

This paper introduces "CubanSea" as a cluster-based visualization of search results. The design strictly follows Shneiderman's principle of "overview first, zoom and filter, then details on demand". Different techniques are combined from existing and novel research for constructing a highly effective design.

Upon searching, the user is presented with a clear overview outlining the topics the result space consists of. The identification of these topics is aided using color encoding and an automated topic header generation. Viewers can then explore the different topics using a dynamically growing column-based list, thus resolving the need for pagination and by filtering this list.

1. Introduction

The primary medium used for retrieving information from the web has always been a list. In the early days of the internet, Sir Tim Berners-Lee maintained a list of all available web pages that could be used for identifying possible sources of information. Search engines replaced that list as the amount of web pages became too large to be centrally maintained. These engines, too, display the results as lists.

The downside of this list-based visualization is the limited amount of information that can be percieved and processed by the viewer simultaneously. Usually, search engines try to provide a ranking of all search results, ordering the result list accordingly. The only domain used for encoding metainformation is the horizontal position of search results inside the result list. Furthermore, in order to achieve a performance increase, the list is split into different pages that the user can browse through. This restricts both the total amount of results and the selection of possible results that can be compared directly to one another.

Search engines are a necessity for performing information retrieval on the web. Most tasks undertaken online require at some point the use of a web search engine, making them the most frequently visited pages online. Hence, the success of most of these tasks, greatly depends on the quality of those search engines. Consequently, a significant amount of research has been undertaken trying to improve their visualization technique. According to [1], the basic principal for visualization is "overview first, zoom and filter, then details on demand". This basic principle can be considerd crutial for the success of a visualization. Unfortunately, no research undertaken so far that successfully implements this principle. It can be hypothesized that this might be one of the reasons why until now no design has been widely accepted. The primarily used web search engines Google, Yahoo, and Microsoft Live still use traditional result lists [2].

One major concern of visualization is the ambiguity of the result space. Users provide a search query that is used for identifying which results are relevant. This query, however, might refer to several distinct topics. The search term "piracy", for example, can refer both to modern licence transgressions regarding digital propety and to historic sea robbery. Naturally, search engines are unable to identify in which one the user is interested. Consequently, result items from both topics are being returned in shuffled manner. Traditional systems suggest that the user refines his query until he achieves a homogeneous result space unambigously identifying the desired topic. This, however, requires the user to have a preexisting knowledge about what topics are applicable to his query and to execute additional network requests.

This paper presents an alternative interface for search engines that visualizes the ambiguity of the result space while applying Shneiderman's priciple. The topics that the result space consists of are displayed, allowing the user interactive exploration.

2. Related Work

As noted above, a significant amount of research has already been undertaken in the area of search result visualization. Most projects propose designs that position result items next to each other according to their simmilarity. [3] suggests a bookshelf-like design, creating different areas for different topics. Result items are placed in accordance to their membership in these topics. The bookshelf is created on demand using a self-organized map. [4] proposes to position search results on a two-dimensional canvas, encoding the simliarity of results in the distance between them. For this, a multi-dimensional vector is created for each result, consisting of the word frequencies of the corresponding document. This multi-dimensional vector is then be projected onto a two-dimensional plane. [5] uses a similar technique, projecting the vector into a three-dimensional space. Areas of result accumulation can be interpreted as distinct topics by the user. All of these techniques require the user to deduct the semantic meaning of these topics by examining the results contained. Furthermore, they position documents that belong to more than one topic in the middle between those. This, however, becomes hard to decode as the number of topics exceeds the number of dimensions available for positioning.

Another encoding technique frequently suggested is the use of connecting lines to visualize when certain documents are related or connected to one another. [6] is one of the papers suggesting this. It calculates the relationships between documents using their similarity.

A number of papers suggest using color to encode the relevance a result has to a search query. [7] splits the query into its different terms and assigns each a specific color. An icon is presented for every result that encodes the relevance of the result to its most relevant term using color saturation. A similar technique is proposed in [8]. Conventional search engines use positioning to encode this information.

All of these designs have one thing in common: they either display the entire result space at once or use pagination and present the top X results to the user. The visualization presented in this paper solves this problem. It is primarily influenced by [3] and [8] and tries to leverage their findings. Furthermore, it provides a solution for automated topic generation and aided topic recognition through the user by generating topic headers. The problem of assigning results to multiple clusters is solved by using fuzzy clustering.

Several papers discuss means of refining the the result space. [9] suggests aided refinement of the search query by analyzing what other terms are frequently used in the result space. This technique requires the user to resubmit his query, thus impairing the system's performance. [7] proposes a card design, enabling the user to select subsets of the result space. Unfortunately only one of those subsets can be displayed at a time and the subsets are automatically generated to be disjoint without giving the user any option of altering them.

A simple and more efficient approach is to filter the result space on demand using a filter query. This allows the user to refine his query and to extract relevant result documents without requiring additional network requests. The result list can simply be minimized to the subset of all results containing the filter query. Since this query can be freely defined by the user, any kind of subset can be adaptively created on demand. The design in this paper presents such a solution, integrated into the cluster-based visualization.

3. The CubanSea Visualization

3.1. Overview: Result Clustering

CubanSea is the name chosen for the implemented visualization interface prototype. It abbreviates "cluster-based visualization of search results". Instead of an ordered result list, the CubanSea interface returns a set of distinguishable areas, each containing a subset of the total result list. These subsets are overlapping and cover the entire total result list. Each area corresponds to one topic occuring in the result space. Figure 3.1 displays the visualization generated when searching for "piracy". As you can see, two different topics have been identified: "software pirates report" obviously refers to software piracy, or piracy of digital media in general, while "pirates history robbery" talks about historic sea robbery. These topic headers have been automatically generated and provide the viewer with the ability to immediatly grasp the topic space. Aiming to increase percivability, each topic is being encoded with a distinct color. These colors have been chosen in accordance to [10], identifying green, red, blue, and yellow as being the most effective color set for labeling. Since yellow has a very low luminance contrast to the white background, it has been replaced with a dark grey. In order to prevent using high volume colors on large areas which would distract the viewers attention as pointed out in [11], a light gradient has been chosen. This gradient is enough to convey a basic understanding of the cluster colors to the user without taking the attention from the content. The top 20 results of each cluster are provided to aid the in topic recognition in the event that the headers might

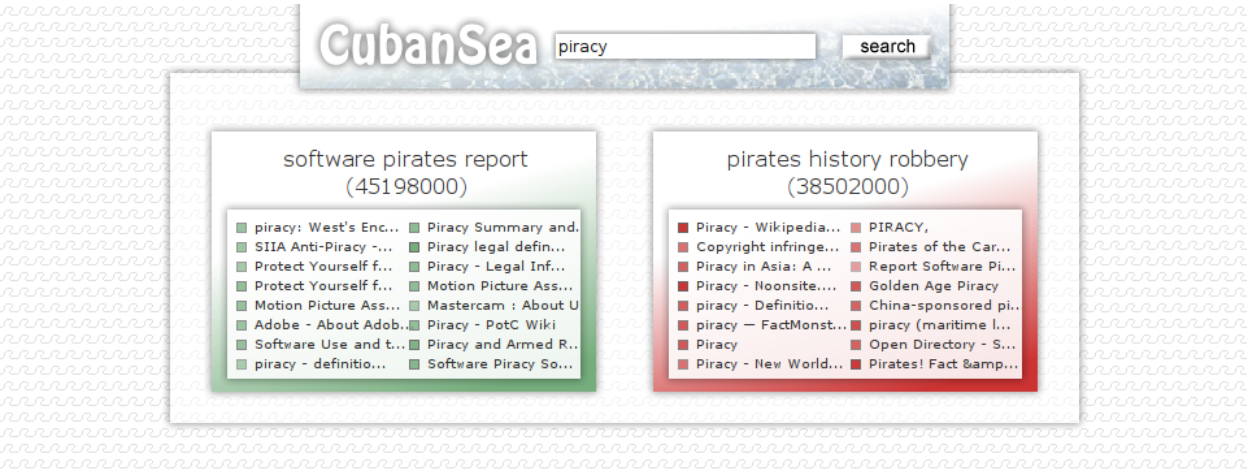

Figure 3.1. Clusters generated for the query “piracy”

be ambiguous. The results are printed on a white-faded background to counter the interfering effect gradient background has on text.

The CubanSea interface only provides the visualization of search results. The results themselves are being retrieved from a search engine provider. For the purpose of this paper, the Yahoo Search API [12] has been chosen. However, the system is designed to work with any search engine. In order to generate the visualization, an initial amount of search results has to be retrieved from the search engine provider. The number of this initial retrieval is critical to the success of CubanSea. As mentioned before, performance is a vital aspect for search engine interfaces. Fetching results from the search engine provider requires a number of network connections, thus posing a bottleneck for the entire application. The fewer search results are initially retrieved, the better the performance of the system. However, clusters construction yields better results when a larger data set is being used. Hence, it is necessary to balance number of initially requested results to achieve the optimal mixture of performance and cluster quality. During my experiments, 50 results has proven to be a reasonable amount.

After retrieving the initial search results from the provider, it is necessary to convert these results into multi-dimensional vectors that can be processed by clustering algorithms. [13] provides a successful solution for this, using the *Porter Suffix Striping Algorithm* presented in [14] to calculate the number of occurrences of different term-stems inside the result documents. [3] suggests additionally multiplying these occurrences with the inverse document frequency. However, applying this proved no significant increase in the quality of the results. I therefore decided against it, favoring performance increase. Search engines typically return only a short summary of the result documents, called “snippets”. Retrieving all original documents prior to performing the clustering would require 50 network connections. Since this would be a significant lack in performance, it is more feasible to use the snippets for constructing the multi-dimensional vectors. Initial experiments to also consider the title of the document resulted in unsatisfactory results. It can be hypothesized, that this is due to the fact that too many titles use the same set of words such as “overview”, “home”, or “results”. Snippets, however, display the actual context of the term’s utilization inside the document. Hence, it is likely to be more relevant and meaningful in respect to the query.

For the clustering algorithm I have chosen the *Fuzzy-C-Means Algorithm* discussed in [15] and [16]. This algorithm iteratively calculates a predefined amount of centroids in a multi-dimensional space and evaluates the membership of the individual items to these centroids using the formula in equation 3.1 where u_{ij} is the membership value of result i in cluster j. While x_i represents the multi-dimensional vector of result i, c_j holds the vector representing the centroid of cluster j. m is a so called “fuzzifier” determining the crispness of the clusters. [17] shows a fuzzifier between 1 and 2.5 to be most effective. For this application, 1.5 has proven to be a reasonable choice. $|c|$ denotes the number of clusters.

$$u_{ij} = \left(\sum_{k=0}^{|c|} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}} \right)^{-1} \tag{3.1}$$

Generally, any kind of distance function can be used

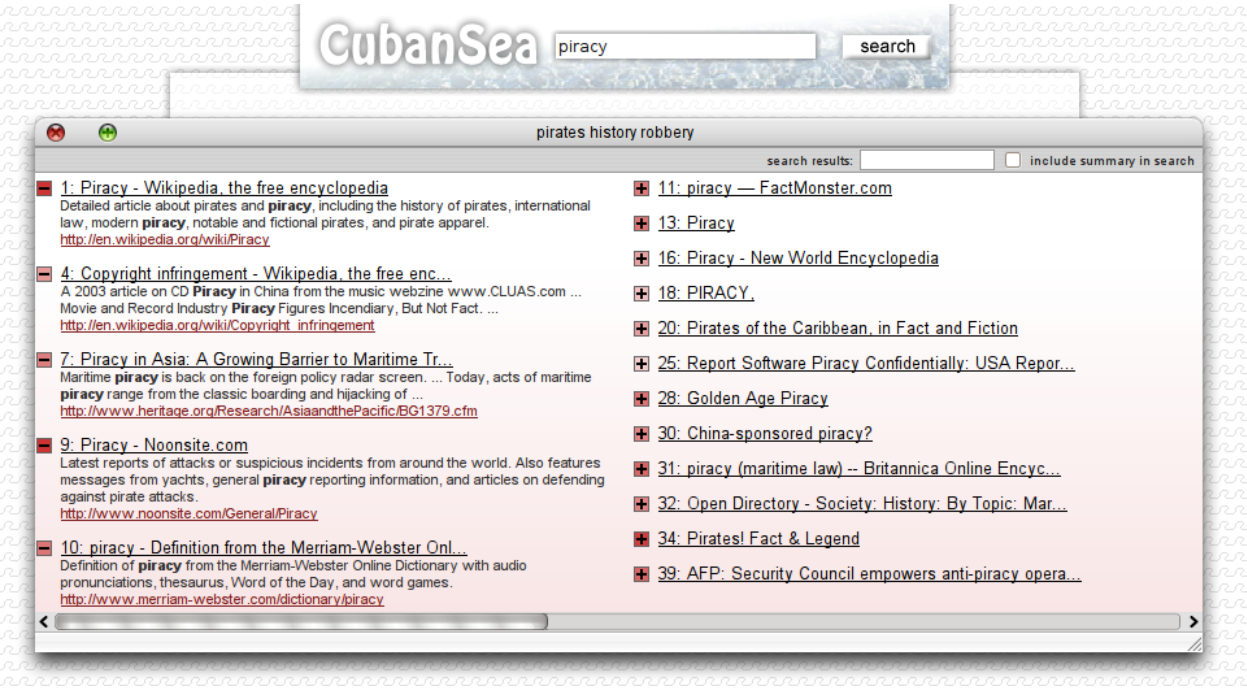

Figure 3.2. Displaying the “pirates history robbery” cluster

for $\|x_i - c_j\|$ in this equation. Due to simplicity, the eucledian distance was chosen for CubanSea. Each iteration optimizes the centroids and recalculates the membership values. As soon as this optimization does not yield a significant change, the algorithm terminates, returning the current centroids as the resulting centers of the clusters.

The primary goal of this clustering is to provide the user with some kind of overview over the result space. This overview is helpful since the human mind is limited in how many objects he can preattentive process [18]. Obviously, it would be counterproductive to generate more clusters than that number. My experiments have shown four to be an effective number of clusters to be generated by the algorithm, lying well below the suggested maximum of 5-10. Clearly, it cannot be assumed that the result space always consists of exactly that many distinct topics. Hence, by configuring the clustering algorithm, we intentionally restrict the visualization to display only the four most relevant clusters. In case that there are actually less than four distinct topics, the cluster centroids will lie close to each other. By simply merging all clusters with a distance to each other less than a certain minimum, the cluster set can be reduced to those actually representing distinct and meaningful topics.

After generating the different topics, the results now have to be assigned to the different clusters. For this, a minimal membership is defined. Results are assigned to all those clusters in which they have a membership equal or higher to the minimal membership. In case that a result is not assigned to any cluster this way, it is simply added to the cluster it has the highest membership in. The number of total results for a cluster as displayed in figure 3.1 can be interpolated by multiplying how many percent of the retrieved results are assigned to a cluster with the estimated total number of results provided by the search engine provider.

Finally, the topic headers must be determined. The headers are constructed using the most frequently used keywords occuring in the topics. Each cluster centroid consists of a multi-dimensional vector recording the number of occurrences of the different word stems. Identifying the three most frequently occurring stems is trivial. However, word stems are no regular words and thus using them can be obstrusive. This problem can be resolved by determining for each stem the most frequent word containing the stem inside the result documents. Naturally, the search terms should be ignored in this process since they are likely to appear frequently in all clusters. Their use would diminish the unambiguity of the headers. This process does not guarantee to generate unambiguous topic headers, but provides a good starting point and the results are sufficient for this application.

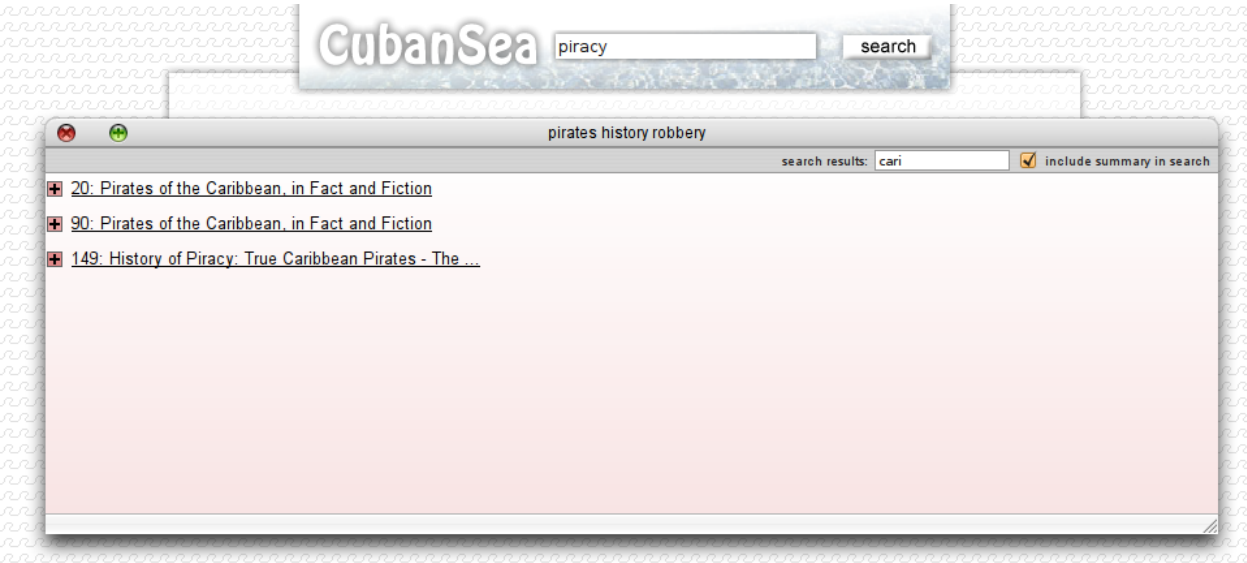

Figure 3.3. Filtering the result space using “cari”

3.2. Zoom: Column-Based Lists

The next step identified by [1] is to zoom in order to explore areas of interest. Upon clicking on the clusters presented in the overview, CubanSea opens a new window and displaying the result list as depicted in figure 3.2. The content of a cluster is displayed similar to a regular result list. This is intentional in order for users to feel more comfortable working with it. The primary difference is that instead of a pagination, a dynamically growing column-based list is used. The total result list is displayed split into multiple columns using a fixed height. Result overflow is handled by adding additional columns, thus changing the scrolling direction from horizontal to vertical. This technique allows users to simultaneously percieve a high number of result items and is based on traditional layouts common in file browsers such as the Windows Explorer and Nautilus. Consequetly, viewers are largely familiar with this visualization and intuitively work with it.

Instead of requiring the user to issue a “go to next page” command, the interface continously fetches additional results from the server and adds them to the list, while allowing the user to simultaneously work with those results already fetched. As a result, the user can navigate through the result space more quickly, simply by scrolling to the right and left. Potentionally all results can be viewed within one single interface. The downside to this technique is the continuous server load. As long as results are available, requests are issued fetching these results. Traditional pagination interfaces only fetch specifically requested resources. For this application, in consideration of increasing hardware and network performance and due to its experimental nature, this downside is acceptable. Future scalability could be easily achieved by evaluating the users scroll behaviour, requesting new results only as the user reaches the end of the currently displayed list and thus implicitly requests additional results.

The column-based result list is enriched with small squares encoding the relevance of the result to the cluster as discussed in [13]. However, instead of using a color scale comprised by distinct colors, the primary color assigned to the cluster is used and its opacity is set to the relative membership value. The relative membership value is the membership value of the specific result divided by the highest membership in the cluster. This way a continuous scale from 0.0 to 1.0 is achieved with 1.0 being the hightest membership, thus being most relevant to the cluster. By using the opacity, the squares of relevant results have a higher contrast than those of results less important. Hence, they stick out more [11]. Consequently, the more relevant a result is for a cluster, the more attention it attracts with its square, hereby aiding the user in identifying interesting results. In order to enhance the perception of differences between high rated results, the relative membership is taken to the second power.

3.3. Filter: Result List Reduction

Especially when dealing with large result lists as those likely to be yielded by omitting pagination, identification of interesting topics can be complicated.

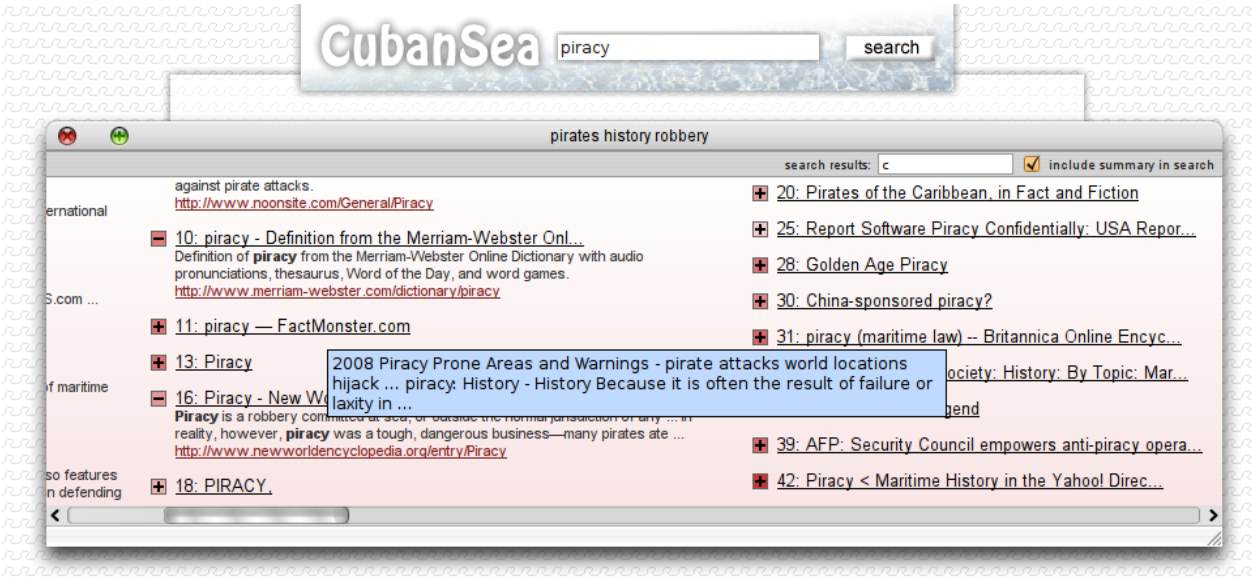

Figure 3.4. Using the mouse-over tooltip functionality for displaying the snippet

[1] proposes filtering the visualized data. The straightforward approach for doing so is to provide a search input field that, upon change, filters the result list using JavaScript. Since all filtering methods are executed asynchronously on the client, the user can interactively redefine filters as he likes without having to wait.

By adding this functionality to the dynamically growing column-based result list, the user is facilitated with the means to quickly identify all results that might be of interest to him. Figure 3.3 displays the result of entering the search phrase “cari” into the input field. The intend of this example is to find all results somehow talking about the piracy in the caribbean. The result list is interactively reduced to those items containing the phrase “cari”. By checking/unchecking the checkbox “in summary”, the user can control whether the snippets shall be used for identifying relevant results or not.

3.4. Details On Demand: Collapsable Snippets

The last step of Shneiderman’s principle is to display details on demand. Clearly, giving all information at once is likely to overstrain the viewer. Search engines typically display snippet and url of a search result below its title. This behavior has been modified in CubanSea in order to save space and to allow the simultaneous perception of a larger amount of result items. Instead of displaying all snippets and urls, only those of the first five results are given. The use case study undertaken in this paper shows that most people typically find their answer within those items. Hence, the detailed information of the other items may be initially neglected. The user can manually choose which snippets and urls he wants to be displayed by clicking the large “+” sign inside the relevance square. A “-” hides this information respectively. Furthermore, snippets can be viewed inside tooltips by moving the mouse over the results as displayed in figure 3.4

Beyond these details, every result can naturally be clicked in order to open the result document, just as in any other search result visualization.

4. Evaluation

In order to determine success, applicability, and weak-points of the CubanSea design, a user study has been undertaken consisting of an assessment and an online survey, trying to compare CubanSea to traditional search engines. A total of ten users contributed to this study. Six users participated in the verbal assessment, seven in the survey. Two users provided incorrect or incomplete filled-out survey sheets, while five users completed the online survey successfully. Four small tasks of gathering urls of web sites relevant to a specified search query and of extracting information hidden within the documents of the result space were defined. Two groups were created, each performing two tasks with their acquired search engine and two using CubanSea. The time and correctness was measures. Furthermore, users were asked to answer subjective questions on what they thought about the

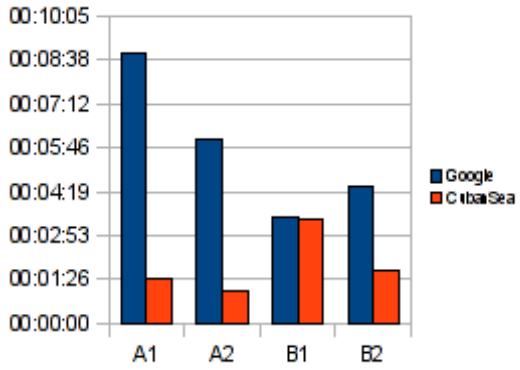

Figure 4.1. Average required time to complete task

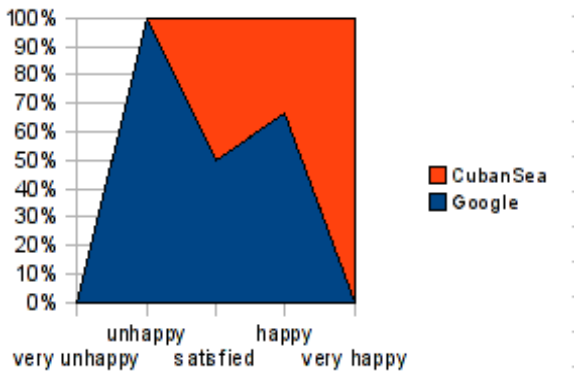

Figure 4.2. User happiness distribution

systems, to assess their quality, and to compare them with one another. Maybe unsurprisingly, all users identified Google as their commonly used search engine.

The primary criteria for CubanSea is the aided recognition of relevant search results. For this, time and correctness of the undertaken tasks are critical. The results gathered by measuring the time revealed a surprising success. Task completion was much faster than when using Google. Figure 4.1 displays the results. Using CubanSea, users required in average only 43.1% of the time they needed using Google for completing the exact same task. This result is specifically interesting since CubanSea is only a prototype running on a low performance machine, thus making CubanSea considerably slower than the speed optimized Google Search. No difference could be recorded in result quality. Only two results were erroneous, both coming from task B1: one was created using Google, one using CubanSea. Further studies would be helpful determining the individual impact on these results for the different technologies used in the CubanSea visualization.

Besides collecting empiric data, the users also had to answer questions on their subjective satisfaction with both Google and CubanSea, as well as compare the two engines with one another. This subjective happyness is crucial for the success of a search engine. No visualization can ever hope of being successfully accepted if users are not comfortable using it. The results were inconclusive. Even though CubanSea recieved in average better ratings, most of the answer space overlapped. Users exist both liking and disliking either one of the interfaces. No connection could be discovered between user preference and personal user data such as age, gender, cultural background, occupation, or experience with computers. Figure 4.2 displays the result distribution for the question “how happy are you with Google / CubanSea”.

During the assessments, users were guided through the process of using the interface application. Those users participating in both the assessments and the survey took the survey first. Users were asked to perform two tasks of their choice from the survey by using CubanSea. While doing so, their actions were observed and feedback was gathered. Three of six users initially wanted to reformulate their query. One user had trouble understanding the meaning of the different topics presented in the overview. Most users (67%) criticized the performance of the system which was to be expected due to the nature of the experimental deployment. Otherwise, all users subjectively liked the system. Specifically the filter functionality was highly praised.

Two users criticized the fact that only the first five result snippets were initially visible. They suggested the initial display of all snippets. Interestingly, both users had previously replied to the question on where they would usually find their search results with “within the first few results returned by a search engine”. It would have been expectable that those users wouldn’t need the other results and thus not their snippets. Further studies should be undertaken to evaluate the potential benefit of hiding the snippets from all results but the first five.

Users also criticized that the clustering algorithm did not prevent result items from being assigned to the wrong topic. In order to correct this, additional research is required optimizing the clustering algorithm. The general visualization of the different topics, however, was accepted. Errors were encounterd in the automated collapse of clusters. Regularily, clusters that obviously deal with identical topics are not discovered to be the same, but are maintained as two separate topics. Also, the topic header generation sometimes yields ambigouos topic names.

The overall result of the use case study was very pos-

itive and encouraging. The visualization successfully aids in the process of performing search tasks, reducing the required time without compromising the result quality. The interface recieved in average higher ratings regarding the user satisfaction than traditional search engines and no critical weak-point was discovered.

5. Conclusion and Future Work

CubanSea successfully applies Shneiderman's principles of visualization while providing a solution for topic recognition in the result space. The fuzzy-clustering algorithm together with the automated generation of topic headers facilitates users with the means to extract semantic information about the result space and to restrict their result space to those results relevant to their search. Specifically the filter functionality is very successful as a tool enabling the user to query the result space and extract subsets of it.

However, still a significant amount of issues are not dealt with. Problems were encountered during the user studies in the algorithms generating the clusters, assigning results to the clusters and generating the cluster headers. Further research is necessary in order to optimize these algorithms. It must also be evaluated whether or not the display of only the first five snippets is sufficient. Performance is obviously another factor. Different deployment methods must be examined and evaluated. Also, the prototype has to be optimized in order to achieve a highly performant application.

Since all tests were based on tasks specifically designed to compare CubanSea with Google, the results might be not transferable to real-life scenarios. Long-term testing on a broader user base is required in order to evaluate the real effectivness and quality of the system. Additionally, the reuse and analysis of selected result sets should be empowered. A prototype extension providing this was planned for this paper but dropped out of scope.

References

[1] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proceedings of the 1996 IEEE Symposium on Visual Languages*, 1996, p. 336.

[2] Nielsen//NetRatings, "Nielsen netratings search engine ratings," http://searchenginewatch.com/2156451.

[3] A. Rauber and H. Bina, ""'andreas, rauber'? conference pages are over there, german documents on the lower left...": An 'old-fashioned' approach to web search results visualization," in *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, 2000, pp. 615–619.

[4] F. Paulovich, R. Pinho, C. Botha, A. Heijs, and R. Minghim, "Pex-web: Content-based visualization of web search results," in *Proceedings of the 12th International Conference on Information Visualisation*, 2008, pp. 208–214.

[5] K. Einsfeld, S. Agne, M. Deller, A. Ebert, B. Klein, and C. Reuschling, "Dynamic visualization and navigation of semantic virtual environments," in *Proceedings of the Tenth International Conference on Information Visualization*, 2006, pp. 569–574.

[6] C. M. Zaina and M. C. C. Baranauskas, "Revealing relationships in search engine results," in *Proceedings of the 2005 Latin American Conference on Human-Computer Interaction*, 2005, pp. 120–127.

[7] S. Mukherjea and Y. Hara, "Visualizing world-wide web search engine results," in *Proceedings of the 1999 IEEE International Conference on Information Visualization*, 1999, pp. 400–405.

[8] O. Hoeber and X.-D. Yang, "The visual exploration of web search results using hotmap," in *Proceedings of the 2006 Conference on Information Visualization*, 2006, pp. 157–165.

[9] ——, "Interactive web information retrieval using wordbars," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2006, pp. 875–882.

[10] B. Berlin and P. Kay, *Basic Color Terms: Their Universality and Evolution*. Berkley: University of California Press, 1969.

[11] C. Ware, *Information Visualization*. San Francisco: Morgan Kaufmann, 2004.

[12] Yahoo Inc, "Yahoo! search web service," http://developer.yahoo.com/search/.

[13] O. Hoeber and X.-D. Yang, "Visually exploring concept-based fuzzy clusters in web search results," in *Proceedings of the Atlantic Web Intelligence Conference*, 2006, pp. 81–90.

[14] M. F. Porter, "An algorithm for suffix stripping," in *Readings in Information Retrieval*, K. S. Jones, Ed. San Francisco: Morgan Kaufmann, 1997, pp. 313–316.

[15] J. C. Dunn, "Fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973.

[16] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[17] C. Stutz, "Anwendungsspezifische fuzzy-clustermethoden," Ph.D. dissertation, TU München, Sankt Augustin, 1999.

[18] C. Healey, "Choosing effective colours for data visualization," in *Proceedings of the 1996 Conference on Visualization*, 1996, pp. 263–270.