Relational browser++: An interface for exploring and searching large information collection

Junliang Zhang
Interaction Design Lab
School of Information and Library Science
University of North Carolina, Chapel Hill
junliang@email.unc.edu

1. Introduction

The size and breadth of digital government information published on the web (such as the huge amount of statistics data generated by wide range of government agencies) makes it difficult for people to quickly grasp what data is and is not available. Consequently, people usually need an overview of the published information to help them decide if it is worthwhile to look further. As they do look further, it is helpful for their searching and browsing to get an idea of what is in the web site and how many items are available. We believe that users of digital government information will be well-served by highly interactive user interfaces that support alternative views of the information partitions/collections/results sets. The Relation Browser (RB) is such a highly interactive interface for people to explore and search useful information. It has been undergoing development and revision for several years and this paper described the functionality and design process for the most recent version, named RB++. The RB++ provides several interface features to facilitate this process, among which are: 1. global overview of the collection and various partitions within it. 2. highly interactive interface for exploring the collection and items within it. 3. highly interactive and flexible searching utility. 4. Tightly coupling these searching and browsing capabilities.

The next section describes the features of RB++ interface, followed by a discussion of previous work. The fourth section presents the interface design process. The paper concludes with a brief summary of a user study.

2. Description of the RB++ interface

RB++ allows users to begin exploring a dataset either by directly manipulating it or by querying it for specified fields. It was written as a java applet. When the applet is loaded by an Internet browser, RB++ presents the initial interface (Figure 1), which consists of three components. The first component is the overview panel (or browsing panel), which displays different facets (or categories) and facet values (subcategories) for the current dataset. For the EIA (Energy Information Administration) application illustrated here, the different facets about the web site are: fuel type, geography, sector and process. Facet values are labeled and represented by graphical bars with different lengths, which visualize the number of web pages associated with them so that users can gain the insight on distributions immediately. One of the important features about the overview panel is that it facilitates direct manipulation exploration of the relationships between different facets. This feature is realized by simply employing the mouse over operation. When users move the mouse over or click graphical bars (we call them focused bars), other bars are updated instantly with the portion of the bar highlighted to show the conditional distribution of focused bars on other facets (Figure 2). The focused bars actually form a query statement. The RB++ interface considers the relationship between the focused bars under same facet as “OR” and those under different facets as “AND”. This feature not only gives the users the power to gain deeper understanding of the collection but also provides query preview abilities to the users so that they can look ahead to result set properties before they commit to the search operation.

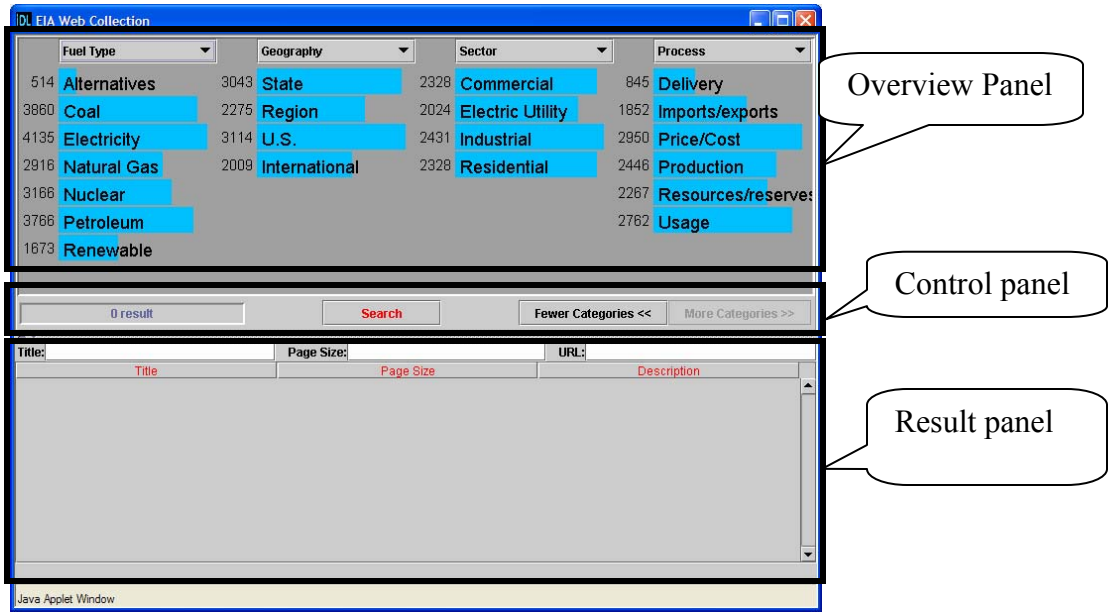


Figure 1. Initial interface with overview of web site

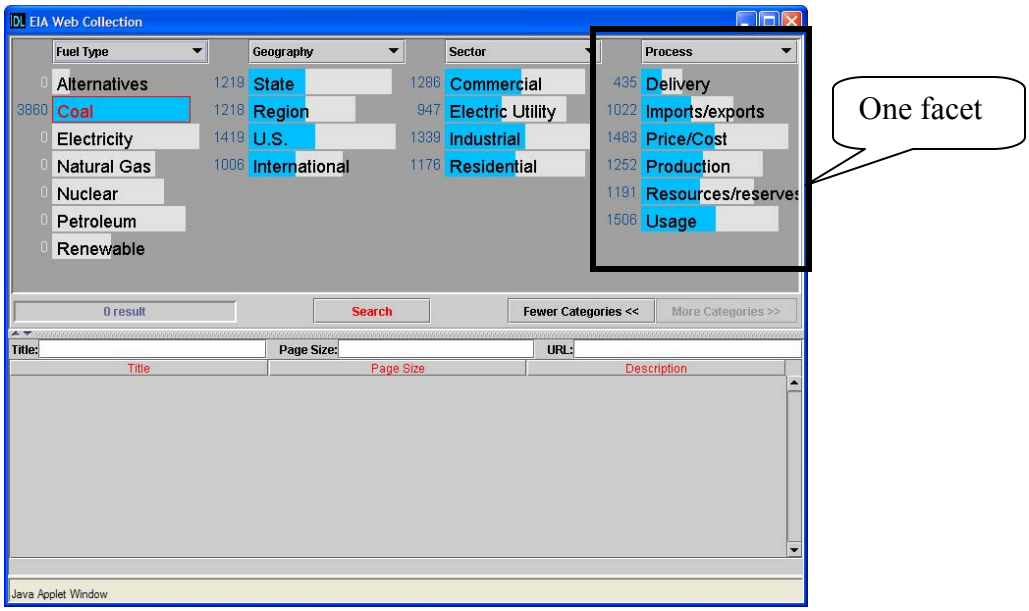


Figure 2. Mouse over “Coal” under the “Fuel type” category reveals the distribution of coal related web pages on other categories

The second component, the control panel, occupies the middle part of the interface. Several functionalities are included on the control panel. Two buttons on the right enable the users to show more or less facets (or categories) in the overview panel. The search button in the middle of the panel enables the users to do the actual query on the focused bars and/or on any keywords typed in the search boxes. The retrieved result set is displayed as a table at the bottom of the interface. On the left is a component to visualize the number of results in the table.

The third component is the results panel which resides at the bottom of the interface. There are several search boxes on the upper part of the results panel, which allow users type textual queries on other metadata of the dataset, such as title, page size, and director.

After users click the search button and get results back in the table, the interface provides rich user interaction allowing users to explore and do further filtering on the results (Figure 3). The results can be sorted by clicking relevant table headings. The sorting can be toggled between descendent sorting and ascendant sorting. Keywords typed in the searching boxes will continuously and instantly filter the results and only qualified ones are displayed. The keywords can be matched anywhere in the text and matched parts are highlighted (Figure 4).

Figure 3. Interface displays the retrieved result of all the "Coal" and "U.S." related web pages. Overview panel is updated as well for retrieved results.

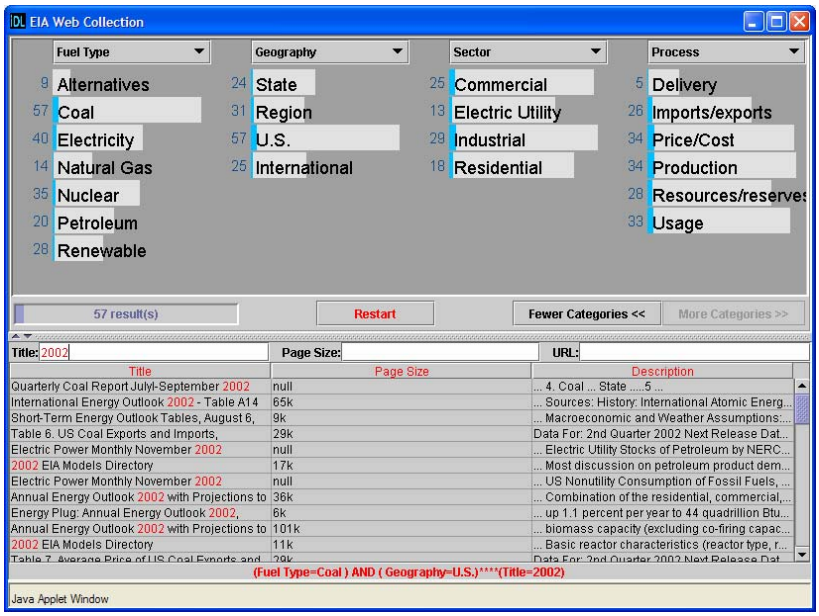


Figure 4. Result table displays filtered results based on keyword typed in search box. Graphical bars are proportionally highlighted for filtered results.

Concurrent with the initial retrieval results presented in the result table, the overview panel also gets updated. The lengths of the bars reflect the retrieved results, thus, the overview becomes

an overview of the retrieved dataset. This demonstrates that different components of the interface are tightly coupled. The search box and the search results are tightly coupled (as described in the last paragraph) and the bars and the search results are also tightly coupled, which means two things: first, mouse operations on the bars such as clicking and/or mousing over also do the filtering on the results; Second, whenever the results get updated, the graphical bars are proportionally highlighted to provide an overview of the updated results, which could expose relationships between the updated results and initially retrieved results. If there is only one result in the table, the graphical bars then inform users exact metadata information about the result. So the bars serve both as browsing components and as searching components.

With the RB++ interface, a common use scenario would consist of four stages:

1. Users could get an overview of the dataset and explore the relationship between different facets within the initial interface.
2. Users employ the bars and/or search boxes to form the query and click the search button to get the initial results
3. Users refine the results within the second interface by browsing and/or searching the initial returned results
4. Users select from among the results those items they wish to display in a new window

3. Previous work

Many information access interfaces aim to provide a starting point for information seekers by presenting overviews of the collection (Hearst, 1999). Overviews can help users understand the whole collections and select or eliminate sources from consideration. Overviews can direct information seekers to specific subcollections quickly, where they can explore details. Usually two types of overviews are employed: category overview and graphical overview. Categories on Yahoo are a familiar example of category overviews. The HiBrowse interface for viewing category labels hierarchically based on facets is another example (Pollitt, et. al. 1994). Another interface using category overviews as faceted metadata is the Flamenco interface (Yee, et. al., 2003). The last two interfaces not only present the category labels to users but also inform the users about the number of documents in each category. However, none of these interfaces allow the users to employ simple mouse over operations to quickly explore the relationships between different categories (or facets). The Flamenco interface could do this as part of the browsing and searching efforts, but it requires a lot of commitment from the user such as clicking the category and waiting. The previous Rave versions (Marchionini, Brunk, 2003) presented various categories and allowed users to explore the relations by mouse over operations, but it only allowed the users to mouse over the main category.

Graphical overviews represent another class of overview, which usually employs various information visualization techniques. Lin (1997) used the Kohenen feature map to visually present a topical overview of the collection. Each block on the map represented a subcollection with similar topics which was labeled by one or two salient words extracted from the subcollection. The adjacent blocks show topic similarity between subcollections. Wise, et al. (1995) developed a three dimensional interface to visually present various topics. Zhang, et. al. (2002) extracted key concepts from the collection and visually presented the concepts in a spring-embeded graph. Similar concepts were clustered together and usually represented a subtopic. The graphical overview is visually appealing, but the usability of this kind of interface has yet to be proven and the 3-D interfaces are more problematic than 2-D interfaces in terms of ease of use and ease of learning.

There has been considerable work on how to present retrieved results in context. Hearst (1996) used a clustering technique to cluster results on the fly and present the different clusters with labeled words to the users to help them understand the results. Chen and Dumais (2000)

employed a classification technique to categorize the retrieved results based on the existing category structure and displayed them in hierarchical categories. These interfaces had to cluster or categorize the retrieval results on the fly, so scaling is problematic. RB++ categorizes the collection off line and uses a uniform category structure to present the overview and the retrieval results. Thus the RB++ can be scaled up easily and reduce waiting time for returning results. We have actually applied the interface to a data set with more than two million records.

There also has been some work on fast lookups for specific information items. Sorting techniques are often used to help users locate the specific item. However, users still need to visually go through each of the items. The Alphaslider (Alhberg & Shneiderman, 1994) is a visual component to help users quickly locate known string items, but it is not very easy to use, especially for novices. Besides, the Alphaslider can only locate the information items based on the first letter alphabetically. RB++ provides an easy and flexible way to locate the information items by inputting the string patterns and the patterns are matched anywhere in the result set. A similar technique was used in the address box of internet browser, but still it strictly matches the pattern with the information items from the beginning.

Dynamic query is a new style of interaction (Shenideman, 1994). It provides a visual interface for the information items and provides visual components to explore the information items by tightly coupling search and visual display of the results. The RB++ was inspired by this work, but instead of providing the visual interface, RB++ employs a more understandable (especially for topical overviews) category structure for the information items. Besides, the search box is a very effective and efficient component for the non-categorized attributes of the items, while the visual component such as sliders or check boxes can only be used for categorical attributes of the items.

The Attribute Explorer (Spence, & Tweedie, 1998) and other similar interfaces (Lanning, et. al., 2000, Wittenburg,et. al., 2001) provide similar ways to explore the relationships between different facets of the classification. These interfaces worked for well structured information. We intend to make the interface work for semi-structured textual information. The search boxes were also provided to help users to put constraints on other types of metadata.

4. Design process

4.1 First design and implementation iteration

The analysis and design of the RB++ began in November, 2002. The work built on the previous versions of the Relation Browser and subsequent RAVE. Two kinds of redesign went on simultaneously: the interface redesign and the backend database schema. The interface of the previous RAVE basically provided a subset of the functions in the new RB++. As in the earlier versions, RB++ displays different categories based on different facets with the distribution shown with visual cues. However, whereas the earlier version only displays one-way relationships, that is, you can mouse over the left most category and see the corresponding distribution on the other categories. (since the other categories were designed to be put in a tabbed pane, you can see the distribution on these categories one at a time). This kind of design makes sense in some situations when the left most category is the main category or mousing on other categories to learn the distribution on other categories does not make sense. However, in most situations, the relationship is reciprocal, meaning that the interface should allow users to mouse over any categories to explore the distribution on the others. In the mean time, displaying all the categories on one panel makes it easy for users to do efficient exploration without extra clicking operations.

Redesign of backend database scheme:

The table storing the data in the previous versions basically was a non normalized table structure, which means that all the relevant attributes of a record are crunched into one table no

matter what the relationship between attributes and the record. Thus, one record can appear multiple times in the table with major part of the attributes unchanged. Besides, non normalized table structure has several problems.

- It is an inefficient storage scheme
- It is very confusing for data structure comprehension
- It does not lend itself toward building extensible, evolving data entry tools

The current database scheme includes several kinds of normalized tables:

- Table ATTRIBUTE_NAMES. This table basically stores heading labels of all the facets (categories) which will be displayed in the drop down menu in the overview panel.
- Table DATA. This table stores the major pieces of information about the data, which are going to be displayed in the results table.
- Table DATA_NAMES. This table stores the labels of the attributes in the DATA table and also the type of data such as number, string etc.
- Tables named (ID)_ATT. These tables store the values for specific facets (or categories), which are displayed on the overview panel
- Tables named (ID)_DATA. These are the bridge tables between the data and various attributes which are used as facets (categories).

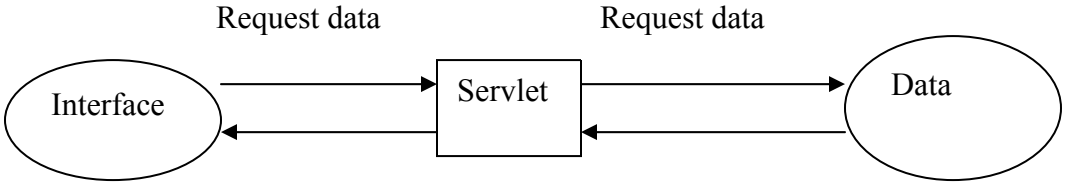
Some of the principles for redesigning the database scheme were:

1. Abstract as much as possible. Numbers are used for attribute tables and data columns. Field names are stored outside of attribute tables and data tables.

2. All relationships between attributes and data are stored as many-to-many, even if there is no need. We cannot know what data is going to be stored in this data structure, but we can assume that much of it will be many to many. Therefore, by creating many-to-many tables between each attribute and the data table, we thus insure that while storage may not be optimized for one-to-many and one-to-one relationships, we can handle all possible relationships.

Overall structure

The overall structure of the RB++ is composed of three parts (basically conforms to the MVC pattern): front end interface, which was written as a java applet; the data repository, which was stored in a Mysql database; the middle layer java servlet, which is used to control and facilitate data communication between the interface and the backend database.



Right now the servlets serve purely as components to complete the data request from the interface. Since the applet cannot communicate directly with the database except when the machine where the applet is invoked is the same one where the database resides, Servlets are mandatory as agents for the applet to request the data from the database. Notice in this structure, the applet class and servlet class must be in the same machine, while the database could be in anywhere.

4.2 Implementation and applications

The implementation of the first revised prototype was undertaken in early February of. 2003. It was written in Java programming language and implemented as an applet which can be invoked from Internet browsers. The first design decision concerned how the mouse over mechanism should facilitate relationship exploration between different categories. Each mouse over action on a category value means doing set of queries on the database. For example, if the user mouses over the "Coal" value under the "Fuel type" category, queries need to be done to return the numeric distribution of the "Coal" related web pages on other categories values. The queries normally can be done on the fly, but tests indicated that there was a 400 to 500 milliseconds delay between the mouse over action and the update of the distribution bar. Most of the delay was caused by the data transfer across the web. There are three strategies to handle the query process: 1. Do it on the fly. 2. Pre-compute the distribution data for all possible queries and preload them into the local memory. 3. A compromised way of first two strategies. The second strategy is not going to scale even for a small dataset, since the number of possible queries increases exponentially with the number of categories or the number of category values. While the first strategy defeats the dynamic query interaction style by causing cognitive interrupts, even though the delay is relatively short. So, the current strategy is to pre-compute the distribution of data for simple queries without those which combine different facets. During the initiation of the applet this pre-computed distribution data is also transferred and stored in the client-side memory. If the potential query is not present in the memory, the traditional way was used to get the distribution data, that is, to do it on the fly. This strategy has the advantage over the first two strategies in that it gives quick response most of time while keeping the preloaded data minimal. One disadvantage, though, is that users may feel some inconsistency in response times.

The first prototype was first applied to the UNC library video collection, given the availability of the collection in a database. The video collection is well structured information, which means it already has different attributes for each entity. In order to apply the RB++ to the structured information, a decision needed to be made in terms of which attributes were going to be put in the overview panel. Only after this decision could the relevant attributes and data be imported into the RB++ backend database. Some important attributes used in the RB++ interface for the video collection were: title, director, genre, decade, and format, but only those attributes which have a limited number of values (such as genre, format) or the value of which can be collapsed meaningfully into a small number of intervals (such as publication year, which could be collapsed into decade). Other attributes can only be displayed in the results panel.

RB++ was then applied to a different domain: the EIA web page collection. This collection is different from the video collection in that web pages are unstructured information and there is no existing metadata (category) available. In order to make the RB++ for the EIA work, a classification scheme for the EIA website had to be generated. The web pages were classified under the classification scheme. While the current classification scheme shown up on the interface was manually generated, we are working on ways to generate the classification scheme automatically.

One of the issues for the RB++ browser is the space issue for the display of the classification scheme in the overview panel when the number of attributes for facets become long. A divided panel java component was used in the interface to partially solve this problem so that users can chose to adjust the space ratio between the overview panel and the results panel. While the space of the overview panel gets smaller, the width of the value bar also shrinks. A so-called fisheye RB++ was also tried to solve the space issue. The Fisheye interface is a focus + context interface, where the objects in the focus area are normal size, while other objects become smaller as the distance to the focus area becoming larger. The code was based on the fisheye menu code from the HCIL at university of Maryland (Bederson's Jazz environment, 2000). The fisheye interface

can handle approximately a hundred values in any facet. However, the usability of the fisheye interface is not clear and bears investigation in future work.

The search boxes were added latter in the first design iteration to help users quickly locate information items in long results tables. The search boxes were provided as a mechanism to manipulate the textual non-categorical attributes for the dataset such as title. In the traditional dynamic query, one of the limits is that there is no component which can be effective and efficiently control the non categorical attributes. The search boxes provide at least a partial solution to this. Many string pattern matching algorithms exist. The search algorithm used in the RB++ interface was the simplest one: brute-force search algorithm, which basically compares the pattern and text from the beginning to the end, if the character doesn't match then go for the next one until find the match or run out of the text. Some other search algorithms were also tried and these trials are discussed in a later section.

4.3. Discount usability test

After the implementation of the newly designed RB++, a discount usability test (Nielsen, 1994) was conducted on the video application. This usability test was meant to find any usability issues on the interface. However, the usability test turned out to be more fruitful. Three people were invited to do the test at the usability workstation in the Interaction Design Laboratory (IDL). No video or audio information were recorded. No time were recorded. But the problems of using the interface to complete the tasks were recorded and a brief post-test interview was conducted. Three types of tasks were designed for the test:

- Find a specific item, where the item could exist or not exist in the collection and the item could be a known item or known partially.
- Find a subset of collections. Some tasks contained an exploration subtask.
- Do exploration (information discovery)

Some important usability problems were found as follows:

- Users tend to find counts by "over clicking." For example: To find how many sci-fi movies in VHS format, users only need to mouse over the sci-fi value under the genre facet and look at the count number beside the VHS value under the format facet. However, users sometimes tried to click both sci-fi and VHS.
- Users had difficulty expressing the range value in the data search boxes
- Users may not know the sorting function of the results table
- The clear button beside the searching box is confusing: whether it meant clear the results or clear the text data in the search boxes.
- Users thought they could combine the bar selection & search box to do the initial query

After this usability test, group members discussed improvement of the interface. It was suggested to combine the bar selection and search boxes to do the initial query. Also it was suggested to dynamically update the bar based on the new sets of the results. After this discussion, a second design and implementation iteration started.

4.4. Second design and implementation iteration

Several interface improvements were implemented in this iteration.

- In the initial interface, searching boxes were allowed to do the search, either in combination with graphical bars or not.
- The overview was dynamically changed when the new results set was retrieved from the initial interface

- The overview was tightly coupled to the results table. Thus, the dynamically updated results caused by the searching boxes also update the graphical bars in the overview panel. Besides, the mouse over operation on graphical bars filters the results in the table as well.

4.4.1 Multi-usage issues:

The RB++ interface is a web application, which means that it should be capable of handling multiple users' requests effectively and efficiently. The multiple requests become an issue whenever communication occurs between the applet and servlets. Servlets, in our case, essentially are the agents for the applet to request data from the backend database. The MySQL database itself is capable of dealing with multi requests at the same time, which is not something we are going to worry about at this point in terms of multi usage issue. However, we need to make the servlets capable of working with multiple user requests. Fortunately, the Java servlet supports multi user requests already. The Java servlet generates separate threads to serve user requests. But we need to be careful when we define global variables in the servlet, since these global variables are shared by all the user requests. There are three servlet agents working for the applet:

- One servlet serves to read the preloaded data (such as category labels, minima set of counts data) for the applet during the initialization stage. The preloaded data were queried from the database offline and were written into a single file as an object.
- One servlet serves to do the query on the fly for the distribution data when the queries on the bar don't exist in the preloaded data.
- One servlet serves to query the database to get the results set

For the first servlet, we need to synchronize the file reading process in case that multiple users invoke the applet concurrently.

5. User study

In the fall of 2003, a user study was conducted to examine the effectiveness and efficiency of RB++ interface compared to traditional search interface (Zhang & Marchionini, 2004). Basically two types of tasks were examined in the study: simple look up task (such as I want to find the movie titled "Matrix") and data analysis and collection understanding type tasks (such as how many movies at 1990s were directed by Steven Spielberg). The results showed that even though there is no significant different between the two interfaces for the simple lookup tasks, there were statistically reliably better results on the more complex searching tasks using the RB++ interface. Additionally, users expressed statistically reliably higher satisfaction levels with the RB++ interface.

6. Acknowledgment

Author wishes to thank Gary Marchionini and reviewers for their valuable comments on the paper. This work is supported by a grant from the National Science Foundation (EIA 0131824).

7. References:

Ahlberg, C. and Shneiderman, B (1994). The alphaslider: a compact and rapid selector. CHI'94, Boston, Massachusetts.

Bederson, B., Meyer, J., and Good, L (2000). Jazz:An Extensible zoomable user interface graphics toolkit in java. In *ACM UIST*, 171-180.

Chen, H, and Dumais, S(2000). Bringing order to the web: Automatically categorizing searching results. CHI'00 The Hague, Amsterdam.

Hearst, M (1999). User interfaces and visualization. In *Modern information retrieval*. Ed. by Baeza Yates, R., and Ribeiro-Neto, B. Chapter 10, ACM Press, New York, NY, 257-324.

Hearst, M. and Pedersen, P (1996). Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results, Proceedings of 19th Annual International ACM/SIGIR Conference, Zurich,

Lanning, T, Wittenburg, K, Heinrichs, M, Fyock, C, and Li, G. (2000). Multidimensional information visualization through sliding rods. AVI'02 Palermo, Italy.

Lin, X (1997). Map displays for information retrieval. Journal of the American society for information science. 48(1); 40-54.

Marchionini, G, and Brunk, B (2003). Toward a general relation browser: A gui for information architects. Journal of Digital Information. 4(1), JoDI

Nielsen, J. (1994). Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. http://www.useit.com/papers/guerrilla_hci.html.

Pollitt A. S., Ellis G. P., and Smith M. P (1994). HIBROWSE for Bibliographic Databases *Journal of Information Science*, 20 (6), 413-426.

Shneiderman, B., *Dynamic queries for visual information seeking*, IEEE Software 11, 6 (1994), 70-77.

Spence, R, and Tweedie, L. (1998). The attribute explore: information synthesis via exploration. Interacting with Computers. 11, 137-146.

Tanin, E, Lotem, A, Haddadin, I, Shneiderman, B, Plaisant, C, and Slaughter, L (2000). Facilitating data exploration with quey previews: a study of user performance and preference. Behaviour & information technology. 19(6). 393-403.

Wise, J., Thomas, J., Pennock, K., Lantrip, D., Pottier, M. and Schur, A. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Proc. of the Information visualization Symposium 95*, pages 51-58. IEEE Computer Society Press, 1995

Wittenburg, K, Lanning, T, Heinrichs, M, and Stanton, M (2001). Paraell bargrams for consumer-based information exploration and choice. UIST' 01,Orlando FL.

Yee, K, Swearingen, K, Li, K, and Hearst, M (2003). Faceted metadata for image search and browsing. CHI'03, Ft. Lauderdale, FL.

Zhang, J. Marchionini, G.(2004). Coupling browse and search in highly interactive user interface: a study of the relation browser++. Submitted to JCDL2004, Tucson, AZ, U.S.A.

Zhang, J, Mostafa, J. Hypathy, H. (2002). Information retrieval by semantic analysis and visualization of concept space for the D-lib magazine. D-Lib Magazine. 10.