Visualizing What People are Doing on the Web

Steven P. Reiss and Guy Eddon

Brown University

Providence, RI 02912

401-863-7641, {spr,geddon}@cs.brown.edu

Abstract

What are people currently looking at in their web browser? Do the patterns of pages change over time? Are changes periodic or just related to current events or other factors? We are developing a tool that attempts to provide insight into these and other questions. The tool sits on top of an Internet-scale programming backbone supports large numbers of simultaneous users. The tool itself provides a unique category-based visualization of browsing history and includes the necessary code to obtain the raw data.

1. Introduction

Visualization provides a foundation for presenting large amounts of information to the user within the restricted space provided by today's computer displays. One application of visualization is to understand all aspects involving a software system. Previous work has concentrated on understanding first the internals of a software system and then the dynamic behavior of the system. However, as systems become more complex and intertwined through the Internet, the behavior of large numbers of distributed users of a system over time also needs to be articulated in order to understand the software. Our visualization is an attempt to monitor and view user behavior in real time both to show how it can be done and to demonstrate the utility of the approach. Real time is important here in order to correlate the behavior of users with the behavior of the program.

The concept of a program is changing from a local, self-contained object into an Internet-scale, pervasive, self-organizing, omnipresent entity. Modern systems run on multiple machines, interact over the Internet with large numbers of users, and communicate and share data with other programs. Support for programming and understanding of such systems is minimal.

We are working on an Internet-scale programming system that attempts to address many of the issues that arise in dealing with such systems. As part of this system we created a sample application, *webviz*, that visualizes how people are currently using their web browsers. This application provides useful and relevant information about the use of the Internet and demonstrates that visualization can be used practically to provide information about the behavior of Internet-scale applications.

Webviz gathers usage data from large numbers of users, monitoring the URLs of the web pages they are currently browsing. It summarizes this information by categories and then displays the results so that users can understand browsing patterns over time, can spot trends, and can identify any unusual patterns.

There are four parts to the *webviz* application. The first involves gathering the necessary data. A visualization can only be as good as the data it is based on. For *webviz*, we implemented a monitor that uses the external information provided by various web browsers to determine when and what URLs users were visiting. The data is gathered locally for each user and needs to be centralized in order to be useful.

Displaying raw data about large numbers of URLs is not practical. Instead one needs to summarize the data and visualize the summary. We decided that the best summary of URLs was to group them into categories. We use the OpenDirectory hierarchy to provide the categories (*http://dmoz.org*). The second part of *webviz* then involves taking a URL and finding the corresponding OpenDirectory category. Within each category, we retain information about the number of views, the number of distinct URLs, and the number of users.

The third part of *webviz* involves saving this information and making it accessible to the viewers. The actual visualization of the web data is the final part of the *webviz* system. It uses concentric circles to show different time intervals and shows the available information with appropriate highlighting.

This paper describes the *webviz* system. Section 2. then describes the data gathering and analysis portions of the system. The visualization and user interface to the data is described in Section 3.

2. Gathering Data for *Webviz*

Webviz provides a view of what people are currently looking at on the web. To do this it needs to determine when people browse a page, gather the data from potentially large numbers of users, and then summarize this data so it can be displayed.

We gather the raw data using a history monitor. The monitor can either be run standalone or as part of the visualizer. The monitor looks for history files from the common browsers (IE, Mozilla, Firefox, Opera, Safari) on the various operating systems (Windows, Linux and Mac OS/X). It periodically (every 15 seconds or so) checks if the external history files have been updated. If they have, it determine what has changed since its last check and accumulates the pages. We note that the monitor was a bit tricky to write since the history files, while accessible, are poorly documented (if they are documented at all), and are optimized for access by the underlying tool.

The history monitor gathers the set of web pages for a particular user. The visualizer, however, needs information from all users. We provide a recorder process to accumulate and summarize the data. For each page it sees, the history monitor passes the recorder a random string identifying the user, a time stamp, and the URL. The recorder then summarizes this information by category. Thus, the first task of the recorder is to find the appropriate category given the input URL which is done using a separate tool which operates by building a decision tree based on the Open-Directory hierarchy.

The recorder uses the web page classification to accumulate information. For each category it maintains the set of URLs that were found and a separate set of distinct user id strings. Periodically (around once a minute), it outputs a summary of this information into a file in a global file system, writing one output line per category that has the category name, the number of views, the number of distinct URLs and the number of distinct users of the views for that category. This is the file that the visualizers will eventually read.

The visualizers maintain a data structure that contains summary information for each one minute interval for the past week, about 10,000 distinct intervals. For each interval and category it keeps the number of view, number of distinct URLs, and the number of distinct users. This information is obtained from the data in the global file. Because the recorders only output summary information, the number of distinct URLs and distinct users is an approximation. The system actually keeps track of the total number of distinct items that are reported. It also tracks the number of times these were modified. What it actually reports is approximated from these. If there is only one add the actual numbers are reported. If multiple adds are reported it reduces the number of distinct URLs and distinct users by a fraction dependent on the number of adds and the total number of views.

3. The *Webviz* Display

A sample of the display provided by *webviz* is shown in Figure 1. The display consists of concentric
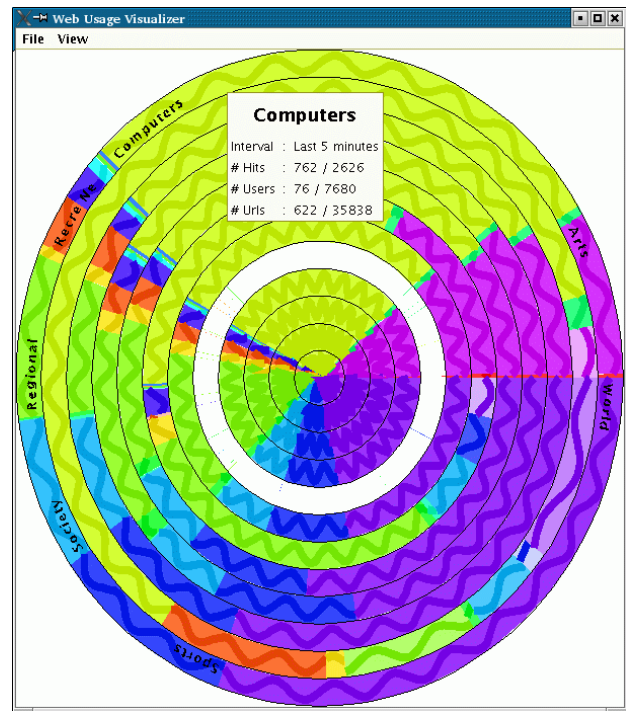
FIGURE 1. The *webviz* display. Each circle represents a time interval. Within the interval each category is given a region with a distinct color. The saturation and brightness of the region and the frequency, width, and amplitude of the interior line code the additional information.

circles, each representing a different time interval, with the outermost interval representing the most recent period. Here the outermost interval represents the last 5 minutes, the next interval the prior 5 minutes, then the last 30 minutes, the prior 30 minutes, the last hour, the prior hour, the last two hours, the prior two hours, and finally the last 8 hours, the last day, and the last two days. A dialog box lets the user set these intervals to start and end at any point during the last week.

To keep the display up to date, the visualizer wakes up periodically (currently about every minute) and recomputes the values for each category. This is done by totalling the saved minute interval values corresponding to each category. Again, an approximation is done here to estimate the number of distinct users and distinct URLs based on the number of intervals summed, the total number of users from the intervals, and the overall totals.

Within each interval *webviz* displays the different categories. These are ordered alphabetically starting at the 3:00 position and going around counterclockwise (i.e. using the mathematical notion of an angle). Each category is assigned a color, with the program choosing the colors initially so as to maximize the differences between successive categories. This is tricky because

the set of categories overall is not known a priori and which categories are going to be adjacent is dependent on the data which is also unknown. Our approach is to divide the hue into a 0-1 range and then to successively divide this range in half and choose categories from different portions of those halves successively.

In any case, the tool presents the user with the legend of what categories are what color in a dialog box that lets the user choose to ignore, merge, or recolor categories.

The span of the category within the circle is based on the number of views, i.e. the total number of web pages for that category during the overall interval as a fraction of the total number of views of all categories during the interval. This lets the user see at a glance what the popular categories are. In addition, because the position of categories remains relatively stable from one interval to another, it is possible to quickly spot when categories suddenly become more or less popular.

We provide labels for categories in the outermost circle. We tried providing labels for all or a subset of the circles, but found that they got in the way. In most cases, we found that only having the outermost circle labeled is sufficient.

While the hue is used to indicate different categories, we also can use saturation and brightness to encode additional information. We have found saturation to not be that useful, but have found brightness, the intensity of the background color within a category region, to be highly visible but nonobtrusive and thus a useful encoding. Here we are using it to encode the relative number of views. This is a measure of the number of actual views recorded for that category during the interval compared to the number that would be expected given the full one-week history of the recorded data. The latter is computed by looking at the size of the interval and the total number of views of the category. A more intense background color indicates significantly more views than expected; a very light background indicates significantly fewer views than one would expect.

For each category in each interval we had additional information on the users, views, and the number of distinct URLs. To code this information we use the line that is visible in the middle of each category region. This line is drawn in full intensity using the category color so it matches the category background but still stands out. We can encode up to three pieces of information with the line. First, the width of the line indicates the relative number of distinct users. This is a measure that compares the number of distinct users recorded for the category during the interval with the number expected given the length of the interval and the total number of views. While most lines have a "normal" size, lines that are particularly fat or particularly thin stand out and highlight those cases where there are much larger or smaller numbers of users than would be expected.

The second piece of information we encode with the line is the relative number of URLs. This is encode using frequency. A very wavy line indicates a higher number of URLs than one would otherwise expect; a flat line indicates a lower number of URLs. Again, lines exhibiting these extremes are highly visible and tend jump out of the diagram to show themselves to the user.

Finally, we are able to use the amplitude of the line to show an additional value. We currently do not use this for any statistic since it tends to get convoluted with a combination of the width of the line and the frequency — any two of the statistics are easy to distinguish, using all three makes the visualization harder to interpret.

A separate dialog box lets the user control the what statistics are displayed with what graphical properties. This lets the user change the focus of the display if they are interested in a particular characteristic of web usage.

The user can interact with the display in two ways. First, we provide tool tips that change as the user moves across the diagram. As seen in Figure 1, the tool tip for the mouse at a particular point specifies the category, the time interval, and the statistics associated with the category during that time interval.

Additionally, the user can click on a region of the display or on a line in the category legend dialog to highlight all the element in the display of a given category. When a category is selected in this way, all the regions for that category are outlined in black, providing the user with an intuitive view of sense of how pages of that category have been viewed over time.

4. Conclusions

The *webviz* application and the TAIGA framework are available for download at *http://www.cs.brown.edu/people/spr/webview.html*. The application itself is reasonably small, with about 800 lines of Java code in the three classification implementations, 400 in the recorder, 300 in the manager, and 3,000 in the visualization program itself.

In conclusion, *webviz* demonstrates that it is possible to collect and display information about program usage or user behavior at Internet scales. Using an appropriate framework, collecting and summarizing the information is straightforward. Moreover, our display techniques demonstrate new and interesting ways of providing meaningful information to the user.