Visualizing Web History

Sheta Chatterjee  
Stanford University  
sheta@stanford.edu

Scott Chung  
Stanford University  
soohark@stanford.edu

Ryan Hamerly  
Stanford University  
rhamerly@stanford.edu

Milinda Lakkam  
Stanford University  
milindal@stanford.edu

Abstract—In this paper, we have explored the use of voronoi treemaps to visualize personal web history. We have developed a novel way of creating a network of websites visited in the past where the edge between two websites is weighed as a function of the difference of the time stamps at which the websites were visited. We run a clustering algorithm on the weighted network and group the websites according to the clusters generated. We then use the voronoi treemap to visualize and interact with the clustered websites.

I. INTRODUCTION

In this day and age, everything has to be personalized. Personalized shopping, news and movie recommendations. Personalized web browsing based on the web history hasn’t received enough attention due to privacy concerns. The current personalized web browser is restricted to showing a select few most visited websites. In this paper, we come up with a novel way of viewing personal web history. The plugin that we created uses the web history to come up with an interactive voronoi treemap visualization which gives an insight into the users browsing patterns and also helps to browse the web using your past web history.

It is very interesting from an individual point of view to discover patterns from web browsing history. A very productive session could mean a cluster of work related sites. A video watching session could mean a cluster of websites which host TV shows and movies. There has been previous work done on giving the statistics of the user’s browsing patterns and creating node link graphs but none of those offer a useful starting point for browsing or show the user such patterns.

The inspiration behind our idea was spawned by an xkcd comic “Map of the Internet” which separates websites geographically based on the volume of daily activity data gathered throughout several months (see Fig. 1).

Having a personal browsing visualization that can be used to explore and visit websites has not been looked at in detail in the data visualization space. Our “Personal Web Mapper” attempts to solve this problem.

II. RELATED WORK

There has been little work done in visualizing personal web history mostly due to privacy concerns. Google chrome has an extension “Visual History for Chrome” (see Fig. 2) that creates a node link graph from the user’s web history but the nodelink graph does not provide much insight to the average user and does not include any interactivity. It was created for the “scientifically inclined as stated in the description of the extension and hence narrows down the user base.

Another extension created for Chrome is the History Analyzer (see Fig. 3) that summarizes the user’s browser history in a pie chart. Each slice of the pie represents the percentage of hits for a domain. The color coded pie chart tells the user at a glance the domain which has been getting the most attention. This extension is meant to be a general overview and again, is not interactive.

BBC News’ visualization of people’s browsing pattern, called Visualizing the Internet (see Fig. 4) comes close to our idea in the sense that it is a tree map of the website categories. This interactive graphic illustrates the biggest sites on the Internet. The size of each rectangle in the treemap is indicative of the popularity of the website. The visualization is created from data collected in 2010 from users in the UK, Spain, US,
France, Germany, Brazil, Australia, and Switzerland. Broad categories are defined and users can mouse over the individual cells to see particular websites. This visualization was created as part of SuperPower, a season of programmes exploring the power of the internet. It is not personalized to a particular user’s browser history, but is meant to illustrate which sites and categories of browsing are popular across the world.

Another treemap visualization similar to BBC News’ Visualizing the Internet is the “Newsmap” application (see Fig. 5). As quoted by Marcos Weskamp, Newsmap is an application that visually reflects the constantly changing landscape of the Google News news aggregator. Google News automatically groups news stories with similar content and places them based on algorithmic results into clusters. In Newsmap, the size of each cell is determined by the amount of related articles that exist inside each news cluster that the Google News Aggregator presents. In that way users can quickly identify which news stories have been given the most coverage, viewing the map by region, topic or time. Our idea to aggregate the user’s web history is somewhat similar.

III. METHODS

There are roughly four steps involved in generating our visualization of personal web history: Data collection, network generation, clustering and creating a voronoi treemap. (see Fig. 6)

A. Data collection

All web browsers collect a history of the user’s activity, and this history data can be processed by browser add-ons using the browser’s extension API. At a minimum, for each page visited, the browser history data gives the title and URL, the visit time, and the cause of the visit (hyperlink, address bar, refresh, etc.) Additionally, some browsers may cache icons and page contents for a limited number of pages, and make the history database searchable.

We implement a Chrome extension to access the user’s history with the Google Chrome Extensions API, particularly chrome.history.1 Because browser history is fairly stan-

1https://developer.chrome.com/extensions/
and the two nodes it connects are merged into a cluster (C). Clustering algorithm function

A linked-node graph, the links weighted by the correlation which does not depend on how often two websites are visited, inequality allows us to define a correlation function

\[ \text{Schwarz inequality:} \]

\[ w_{ab} = \sum_{ij} e^{-|(t_a)_i - (t_b)_j|/\tau} \]  

where \( t_a \) and \( t_b \) are arrays listing the sites’ respective visit times, and \( \tau \) is a time constant (we use 30 minutes, and also employ a 12 hour cutoff for computational efficiency). This function is maximized when the time increments \( (t_a)_i - (t_b)_j \) are minimized – which happens when the visit histories for sites \( a \) and \( b \) overlap. It is close to zero when all the increments are very large – when there is little temporal overlap between visit times to the two sites.

It can be shown that, in the frequency domain, \( w_{ab} \) is related to an inner product on \( L_2 \), and therefore satisfies the Schwarz inequality:

\[ w_{ab}^2 \leq w_{aa}w_{bb} \]  

with equality holding when the two histories coincide. This inequality allows us to define a correlation function

\[ r_{ab} = \frac{w_{ab}}{\sqrt{w_{aa}w_{bb}}} \]  

which does not depend on how often two websites are visited, only on how frequently, relative to their popularity, they are visited together. Each domain name becomes a node on a linked-node graph, the links weighted by the correlation function \( r_{12} \); see Fig. 7(b).

B. Network generation

The user’s web history is a collection of web pages and visit times. We wish to group these pages into meaningful categories based on the user’s browsing habits. Our algorithm is based on the premise that people tend to focus on one activity at a time while browsing the Internet, so similar web pages should cluster in time. If the user visited a research site at 3:00 on Monday, sites visited between 2:30 and 3:30 will tend to be research sites rather than news or entertainment sites. There will always be exceptions, but with enough data, this tendency will dominate over the exceptions and time history should provide a reasonable categorization most web pages.

Over a period of 30 days, a typical user’s web history contains thousands of unique URL’s, which can be grouped into several hundred unique domain names (“websites”). For each site, the browsing data is used to reconstruct a time series of past visits (see Fig. 7(a)). A time coincidence function is defined as follows:

\[ w_{ab} = \sum_{ij} e^{-|(t_a)_i - (t_b)_j|/\tau} \]  

where \( t_a \) and \( t_b \) are arrays listing the sites’ respective visit times, and \( \tau \) is a time constant (we use 30 minutes, and also employ a 12 hour cutoff for computational efficiency). This function is maximized when the time increments \( (t_a)_i - (t_b)_j \) are minimized – which happens when the visit histories for sites \( a \) and \( b \) overlap. It is close to zero when all the increments are very large – when there is little temporal overlap between visit times to the two sites.

It can be shown that, in the frequency domain, \( w_{ab} \) is related to an inner product on \( L_2 \), and therefore satisfies the Schwarz inequality:

\[ w_{ab}^2 \leq w_{aa}w_{bb} \]  

with equality holding when the two histories coincide. This inequality allows us to define a correlation function

\[ r_{ab} = \frac{w_{ab}}{\sqrt{w_{aa}w_{bb}}} \]  

which does not depend on how often two websites are visited, only on how frequently, relative to their popularity, they are visited together. Each domain name becomes a node on a linked-node graph, the links weighted by the correlation function \( r_{12} \); see Fig. 7(b).

C. Clustering algorithm

The websites are categorized using a hierarchical clustering algorithm. The most-weighted link in the graph is identified and the two nodes it connects are merged into a cluster (A + B → G = AB in the figure). The time coincidence function for the cluster is straightforward to compute

\[ w_{GG} = w_{AA} + 2w_{AB} + w_{BB} \]
\[ w_{GX} = w_{AX} + w_{BX} \quad \forall X \neq G, A, B \]  

so the correlation coefficients need not be recomputed from scratch at every step in the clustering. Once the two nodes have been merged and new weights are found using (4), the algorithm finds the most-weighted link in the new graph, and repeats until the graph has only one node. This allows one to construct a binary clustering tree (Fig. 7(c)), where all the leaves are websites and each node is labeled with an \( r \)-value, the time correlation function of its children.

Our Voronoi diagram works best on trees with 10-20 nodes, not two, so the tree is subsequently “rebalanced” by merging child nodes with their parents whenever a child node is only slightly more clustered than its parent \( r_{\text{child}} < r_{\text{parent}} + \Delta r \), where \( \Delta r \) is a parameter we empirically set to 0.12. The resulting tree is much less deep than the original, and has more children per node (Fig. 7(d)). If the tree has too many children for our algorithm to run efficiently, the smaller children at each node are grouped together. Finally, for simplicity of presentation, we flatten the tree at the first level.

D. Voronoi treemap generation

We implemented a voronoi tessellation algorithm that can work with arbitrary shapes by using a half plane intersection algorithm as described in [1]. The starting points of the centroidal voronoi tessellations are sorted by weight and initialized in a spiral pattern going out from the center to encourage more heavily weighted cells to gravitate towards the center of the visualization.
Fig. 8. original implementaion of the personal web mapper

The half planes are calculated using the distance formula 
\[(x - x_1)^2 + (y - y_1)^2 = \text{weight}\] giving polygons instead of hyper planes. In an additively weighted voronoi tessellation data points can be enclosed in the voronoi cell of another data point. The weights are scaled to prevent this behavior. For each point, we find the closest neighboring point. Then, we take the ratio between the weight of the first point and the distance of these from its closest neighboring point. Once we have done that for all data points, we find the maximum of these ratios, and divide all weights by this ratio.

To calculate the centroidal voronoi tessellation, we use Lloyds algorithm and iteratively compute the tessellation while moving the location of the data points to the centroid of the corresponding voronoi cell. To compute the tree map, we run the tessellation with the clusters as data points. Then, we run the tessellation on each cluster with its children as the data points and the clusters voronoi cell as the new bounding shape.

IV. RESULTS

The final product of our work is a google chrome extension. Once the user installs the extension, an icon appears right next to the address bar. If the user clicks the icon, our program extracts the web history of the user, forms the weighted network as described above and creates the voronoi visualization. The user can see clusters of websites and could navigate to a website by clicking on the region allotted to it in the map. The network of the web history is rebuilt with the latest data every time the user refreshes the page.

Note that the clusters of most frequently visited websites tend to be placed towards the center of the bounding box making it easier for the user to navigate to those pages. This is because we initialized the voronoi tessellations as a spiral with the heavier clusters at the center of the spiral. We also avoided labeling each of the individual regions of the voronoi map to avoid cluttering of the labels.

Several implementations of the visualization are shown in Fig. 8, Fig. 9 and Fig. 10.

V. DISCUSSION

After showcasing our work to an audience, we found many interesting insights about our visualization. Many users enjoyed the Voronoi tree map visualization because they could spatially see which groups of websites were visited more. Also, because the voronoi treemap has polygon tessellations, each of a different size, it was easier to differentiate the boundaries between different clusters, and users could easily distinguish between each of the smaller clusters of websites in the visualization by the nature of the polygon shape being non uniform as opposed to a more uniform squarified tree map.

However, we also noticed users attempting to find further patterns in the visualization. Some users believed that there was some sort of correlation between larger clusters that were adjacent to one another, showing some sort of change in interests over time. For example, if the design category of websites was near the technical category of sites, some users inferred that this meant that design related sites were viewed before technical related sites. Although the positioning of the individual clusters was based on the optimization of the space used, it was interesting to see that this dimension of the visualization had the potential to also bring in more insights about the data and perhaps illustrate a different pattern about the relationship between the individual clusters.

The audience responded positively to the idea of making this visualization the new start page of google chrome to base browsing off of. They acknowledged that the existing browser history start page that has only up to eight most frequently visited websites was not too helpful. Many of them stated that being able to see the websites that they had visited infrequently in the Voronoi visualization was interesting and
useful especially if the link was lost in their bookmarks or a site got buried amongst their browsing.

There was an interesting concern raised by one user that using this visualization as a basis for browsing websites could restrict further exploration of the web. Some users were searching for an exploratory aspect of the visualization, trying to find newer websites based on their web history. This brought some insight into how the project can be extended to provide recommendations based off of past browsing data. For example if a user hovers on a cluster, we could provide recommendations of websites that are related to the websites in the cluster.

VI. FUTURE WORK

From the suggestions and patterns observed from user interaction with the Voronoi web history tree map, we created a list of improvements to further enhance the visualization. One idea is to include the added dimension of images, so when users mouse over a cell in the treemap, they can view an image of this site. A way to extend this even further would be to make the cells themselves images, and adjust the saturation and hue of each image in order to prevent the visualization from looking too cluttered or difficult to use.

Another large area to explore with our visualization is the realm of personalized recommendations. Browser history offers a lot of rich data about a user's interests in many different domains including food, work, social life, activities etc. Using this visualization to extract a user's interests and then customizing recommendations to other websites a user could visit, or pointing them to events or activities that fall under their hobbies, could extend the current visualization into a personal system that helps users narrow down sites and events around them that apply to their specific interests, amidst the large web of the Internet.

We could also add additional interactivity by expanding the clusters on mouse hover to get a bigger view and show additional information of the cluster. In addition, we could also cache the network built from the web history for a faster startup time.

REFERENCES