

Dynamic Aggregation to Support Pattern Discovery: A case study with web logs

Lida Tang and Ben Shneiderman

Department of Computer Science
University of Maryland
College Park, MD 20720
{ltang, ben}@cs.umd.edu

Abstract. Rapid growth of digital data collections is overwhelming the capabilities of humans to comprehend them without aid. The extraction of useful data from large raw data sets is something that humans do poorly. Aggregation is a technique that extracts important aspect from groups of data thus reducing the amount that the user has to deal with at one time, thereby enabling them to discover patterns, outliers, gaps, and clusters. Previous mechanisms for interactive exploration with aggregated data were either too complex to use or too limited in scope. This paper proposes a new technique for dynamic aggregation that can combine with dynamic queries to support most of the tasks involved in data manipulation.

1. Introduction

Current technologies have enabled massive collections of data. Newer and faster algorithms for data analysis are always in demand to harness the flood. If the amount of data can be reduced to a manageable size, then humans can find patterns that automated algorithms may have missed. Dynamic Queries (DQ) is an interactive technique for data exploration [1]. Users manipulate sliders to filter out data. Each slider corresponds to an attribute of the data. A requirement of dynamic queries is that the visualization must keep up with the user's manipulation within 100 milliseconds. Since a large portion of the computer's computation is spent on visualization, when the datasets grow, the time to complete drawing grows proportionately. Thus DQ isn't suitable for dealing with large amounts of data.

Large datasets poses two problems to interactive exploration. One is how to represent the elements on the screen fast enough. Second is if you draw it on the screen, can the user even understand it. Visual occlusion is a problem in general for visualization. If the user can't see the data point, then the time spent drawing the item was wasted. This problem can be solved for small numbers of items. The commercial data analysis package, SpotFire (www.spotfire.com), randomly jitters the data points continuously, so that clusters that occupy the same point can be seen. With larger data sets, the occlusion problem grows more and more pressing. Due to the non-uniform nature of most data sets, many data points will occlude others. The visual representation can then deceive users by not showing clusters that exist in the data.

Aggregation is an effective way of managing large data sets. It summarizes groups of similar data elements and can greatly reduce the number of glyphs that are shown on the screen. Because users can specify how to aggregate the data, the important aspects of the data will be preserved while the dataset size is reduced. Patterns that are hidden within millions of data points can emerge dramatically when aggregation reduces these into thousands of points. Fredrikson et al. [5] explored using aggregated data in conjunction with Spotfire, and demonstrated the uses of different kinds of aggregation using highway incident data. Hochheiser and Shneiderman [6] used aggregation to interactively explore web log data. In their study, the aggregation was done manually through SQL queries, though integration with an aggregation tool was suggested as a future direction.

2. Related Work

Using aggregation and Dynamic Queries together in one interface is not a new idea. Goldstein et al. [2] proposed it in 1994. An interface mechanism called Aggregate Manager (AM) was combined with DQ, which produced a powerful combination (Figure 1). DQ is used to select a subset of the data set; this is transferred over to AM as an aggregate group. AM can then do aggregation on different aggregate groups, and pass the data back to DQ for display. This loop fulfills one of the lacking area of DQ: providing conjunct or disjunctive groups. Using AM along with DQ provides many possible combinations for data manipulation, which is powerful but can be hard for users to understand and fully control.

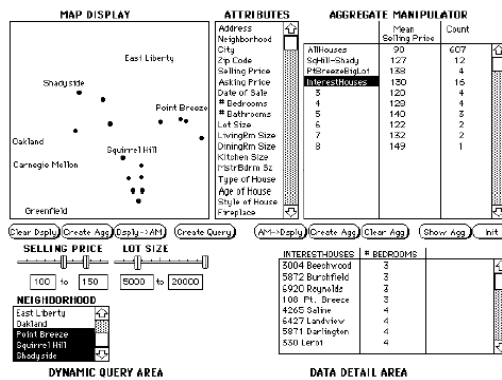


Fig. 1. The workspaces of AM with DQ

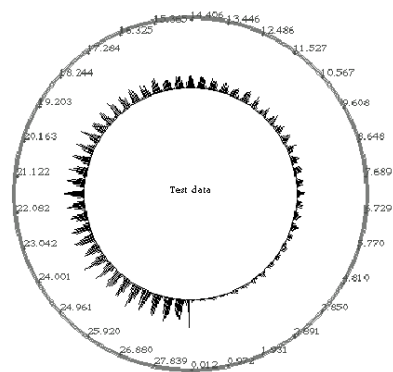


Fig. 2. SolarPlot showing a histogram

An alternative approach to user-controlled aggregation is automatic aggregation. Chuah [3] used automatic aggregation in SolarPlot, a circular histogram (Figure 2). Elements are mapped to a pixel on the circumference of a circle; the height of a spike that emanates from the pixel represents the number of data values that fall within that pixel. This aggregation is intuitive and simple, the scale of the aggregation depends on the diameter of the circle, and the aggregated value is easily understood. SolarPlot

only encode one dimension of data in the visualization, thus any correlations between fields are harder to find.

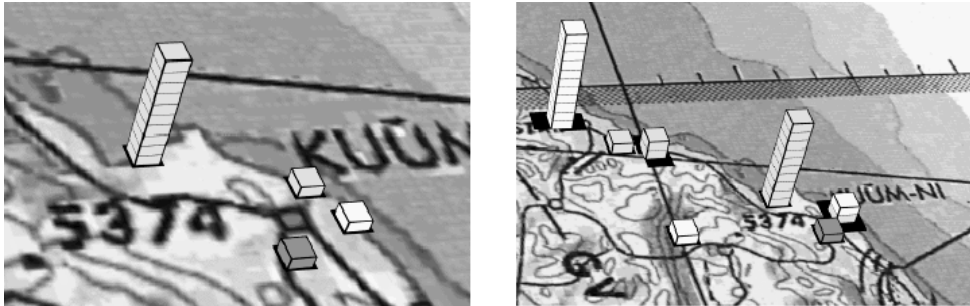


Fig. 3. Close up and zoomed out view of Aggregate Towers

Rayson's [4] Aggregate Towers provide another automatic aggregation interface. Data points are displayed as cubes on a 3d plane. As the user zoom in and out, data points are clustered based on their geospatial location (Figure 3). Stacks pointing out of the plane represent the aggregate groups. The cubes still retain their original color-coding. This automatic technique alleviates 2D occlusion problem by forcing it in to 3D. These stacks of data towers will occlude each other in 3D, but is easily remedied by allowing the user to freely rotate the view.

3. A Simple User Driven Aggregation Interface

Automatic aggregation is useful as a way to reduce occlusion. However, having no user control makes automatic aggregation of limited use for general datasets. Goldstein's AM is complex and hard to learn. A simple interface for user controlled aggregation is a nice compromise.

Spotfire's user interface was used as the starting point of our system. In Spotfire, a scatter plot of two attributes of the data is at center of the screen. Combo boxes at the edges of the axes select the fields being plotted. A panel on the right side displays DQ controls and detail on demand. The entire interface is in front of the user. Our system has similar characteristics as Spotfire. The aggregation controls are located on the left side so that DQ can be placed on the right side. The primary aggregation control is a combo box that can be enabled or disabled (Figure 4). Specifying a group of data manually is easy using DQ. However, creating many such groups can be time consuming and should be automated. The user only needs to select a field to group on, by using the "Group by" widget, and have the program sort out the groups. The default grouping algorithm used is equivalence grouping. For numerical data, equivalence is when they represent the same value, thus 4 and 4.0 are the same and belong in the same group. For categorical/string data, a case sensitive string comparison is used to determine equivalence, thus "4" and "4.0" as string are not the same. Should the user require a different grouping criterion, clicking on the "..." button to the right of the combo box will bring up an options dialog. Here, the user can choose which algorithm to use and to configure the algorithm to their liking. If

the groups that are created are not specific enough for the user, they can be broken down into subgroups. E.g. in the case of census data, we can group the entries based on gender, then subgroup based on age brackets, creating meaningful groups that can be used in aggregation. Subgroups are also controlled by checkable combo boxes. A combo box labeled "Subgroup by" will appear under the "Group by" widget after the user has selected a field to group by.

Once the grouping computation is finished, the results are shown on the screen with each dot now representing a particular group, the size of the dot is currently coded to show the number of elements in that group. The secondary aggregation controls are the aggregate method combo boxes. Those are located below the vertical axes field selector, and to the left of the horizontal axes field selector. The user can select different aggregation algorithms for each axis independently.

4. System demonstration

The dataset used was extracted from web logs. The data is taken from University of Maryland's Computer Science web server. Only the requests that belonged to the HCIL section of the website (www.cs.umd.edu/hcil) were extracted. This is similar to the dataset that Hochheiser and Shneiderman [6] explored in their study. The data have the following five fields:

- Client host
- time: timestamp of request
- url: the URL requested
- return code: the server response code to request
- bandwidth used: number of bytes transmitted for that request

Web log data is very large and has only a few data fields. Traditional web analysis packages create tables of statistics and static graphs. The user merely feed the data to the program, and it is the program that decides what to report back to the user. Hochheiser and Shneiderman argued in their paper that interactive star field visualization, like Spotfire, is a valid way of analyzing web log data. However, in order to find some of the interesting features involved preprocessing and aggregation. Thus, using the same web data will be a good test of the flexibility and power of our simple aggregation interface.

Since the web data consists of individual client requests, one logical grouping would be to group by user. By viewing the size of the groups, one can detect abnormally large numbers of requests from a particular user. We find that the Google spider the most frequent visitor of HCIL. To find out how much bandwidth Google consumed, we change the field we are viewing to "Bandwidth used" and set the aggregator function to sum the field (Figure 4). We found that it isn't Google, but another crawler, EoExchange that is using the most bandwidth. To view the access patterns of the clients, we can subgroup based on the time of access. Figure 5 shows access patterns of users over days. The bandwidth hog EoExchange shows up in this graph as well, while Google's accesses are well hidden and spread out across days.

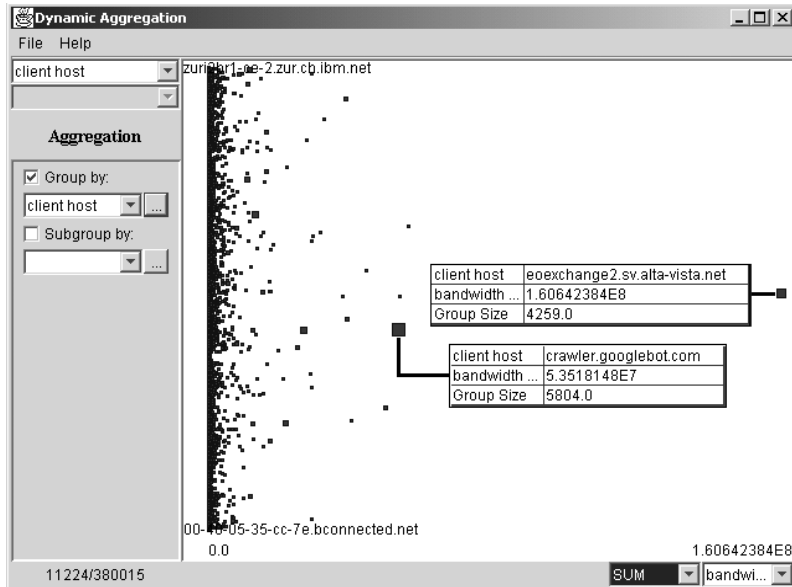


Fig. 4. Finding the most frequent visitor by aggregating by client host

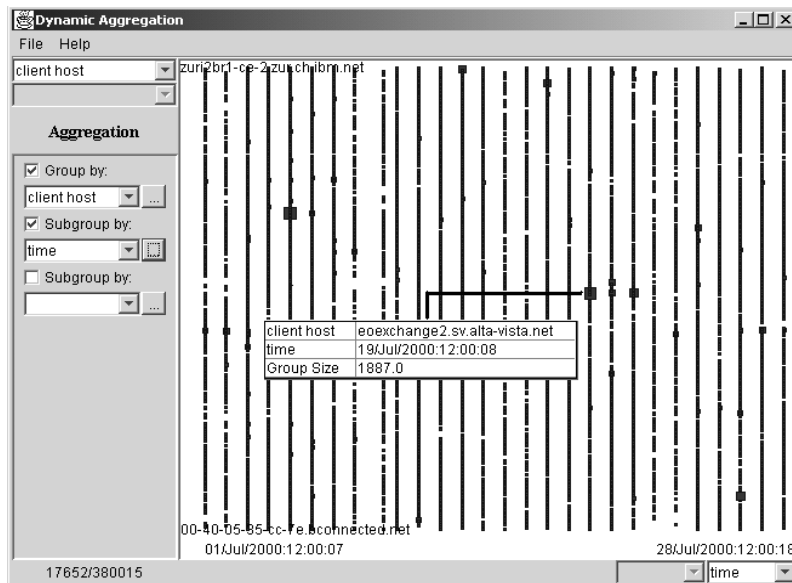


Fig. 5. User activity over days

5. Conclusion

We have developed simple manual aggregation interface that we believe the users can understand and use effectively. However, due to the inherent complicity of the aggregation concept, users should have in mind a specific question they would like answered. Unlike DQ, in which users can explore and experiment with data, aggregation should be thought of as creation of a new dataset. This new dataset can then be explored by DQ. A usability test should be conducted to test how readily users understand using the interface and which grouping algorithm and aggregation algorithm are needed to have a rich set of tools so the user can find answers to more complex questions than what was considered in the paper.

Acknowledgements

We thank Catherine Plaisant for her help in finding related work, Harry Hochheiser for help in getting the web log data, as well as the students of cmisc838b for support and inspiration.

References

1. Shneiderman, Ben. (1994). "Dynamic Queries for Visual Information Seeking." *IEEE Software*. 11(6), 70-77.
2. Goldstein, Jade and Roth, Steven F. (1994) "Using Aggregation and Dynamic Queries for Exploring Large Data Sets" *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems 1994 v.2 p.200*
3. Mei C. Chuah and Roth, Steven F. (1998) "Dynamic Aggregation with Circular Visual Designs" *Proceedings of Information Visualization, IEEE, North Carolina, October 1998*.
4. Rayson, James K. (1999) "Aggregate Towers: Scale Sensitive Visualization Decluttering of Geospatial Data" *Proceedings of the 1999 IEEE Symposium on Information Visualization*
5. Fredrikson, A., North, C., Plaisant, C. and Shneiderman, B. (1999) "Temporal, Geographical and Categorical Aggregations Viewed through Coordinated Displays: A Case Study with Highway Incident Data" *Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation, Kansas City, Missouri, November 6, 1999 (in conjunction with ACM CIKM'99), ACM New York, 26-34*.
6. Hochheiser, H., and Shneiderman, B. (2001) "Using Interactive Visualizations of WWW Log Data to Characterize Access Patterns and Inform Site Design" *Journal of the American Society for Information Systems*, 52(4), February, 2001.