Ross Donatelli, David Campbell
DataViz Lab 3 Grp 11

Part B

Dataset - Stanford
Web data: Amazon movie reviews
http://snap.stanford.edu/data/web-Movies.html

We chose to use the Amazon Movie Review Dataset, because it possesses a few nice properties. First, it has nearly a million users, so we can test our solution against a large number of nodes. Secondly there are lots of interesting questions related to users. We can link users together by using the common movies that each user reviews. If users Alice and Bob both review the movie *Titanic,* then they have a link between them. We can compare the similarity of each ranking of the movie to use as a metric of comparison to determine the edge weight that may define an inverse relationship of distance between the two users. For example, if Bob rates *Titanic* with 5 stars but Alice only gives it 1 star, then the distance should be larger between Alice and Bob.

By grouping users together by similarity of movie reviews, we could then find other users who are similar to a given user and suggest movies to the user based upon what similar users enjoy. Another possibility is that you could use a clustering algorithm such as K-Means to determine group preferences, and when you would like to advertise a new movie that is similar to a genre, you could advertise that movie to a given group cluster.

It will be interesting to see how many isolated graphs we will find. Are there groups of users on Amazon that review movies in relative isolation from the general populace? If so perhaps we identify their tastes so that we can cater specifically to these small groups.

Data Description:
The data provides the following fields in a review:
product/productId, review/userId, review/profileName, review/helpfulness, review/score, review/time, review/summary, review/text.
For our purposes, we don't really need the time, summary, or the text, as doing an analysis on the content of a given review is beyond the scope of this project. We will be focusing on the productId, or the unique identifier for each movie within the Amazon store, the userId, to link users reviews, and the review score, as a metric to compare users. In addition we may potentially score users based upon their review helpfulness to adjust the node size, because reviewers with a high helpfulness have more influence than those who have low scores.

Data Scraping:

To scrape the data we took different sized subsets of the original dataset and then generated two lists, one of nodes and one of edges. Each was encoded in a JSON format, where each node was represented by attributes for it's user id, the count of their reviews, their helpfulness, and their aggregate review score. Encoding helpfulness was tricky, as a typical helpfulness score might be the string '5/7'. Simply using this as a percentage would negate the number of people that found a review helpful. Therefore, we decided to square the number people that found the review to be helpful over the total number of people. Thus helpfulness scales with volume. Edges were created with the source and target nodes, a list of product ids, and a list of weights. Each edge may represent multiple reviews, this was done to reduce the number of edges in our graph. Each weight was obtained by finding the similarity between the two users review scores. To calculate this we took the base review score (5) and took the difference of the absolute value of the difference between review scores.

Unfortunately, we still had issues with explosive growth of edges, even with each edge representing multiple reviews. To combat this, we introduced parameters to control the minimum amount of reviews that each user must have to be represented as a node. Any edge with either user review count below this threshold will be discarded. Initially we set this to 5, but as we incorporate more data, this graph becomes unusable, so it helps to increase this parameter in relation to the size of the dataset. An additional parameter we introduced for the larger set is a minimum number of shared reviews. Setting this to two eliminates the vast majority of edges, which is desirable, because users connected to other users by only one movie are perhaps not as interesting as the more connected users.

Data Visualization

In our graphs, the nodes represent the Amazon reviewers, and an edge between two reviewers indicates that those two reviewers reviewed a product in common.The edge given has two properties that are determined by the data. The first is the width of the edge, which grows thicker the more common reviews the two reviewers had. For example if reviewer A and reviewer B reviewed 6 of the same products, they would have a thicker line than two reviewers that reviewed only 1 product in common. The second property is the distance of the link. This is determined by the commonality that the two users have on a review. If two users have multiple common reviews, the commonality is averaged to create this distance.

In order to visualize the graph, force is applied. Initially the nodes are randomly distributed throughout the svg canvas and force is off. This is done to help increase responsiveness in the web browser for larger graphs. The user then presses 'r' to turn on force, or 't' to tick the force. Ticking then force gives the graph one time step worth of force physics applied. This was also implemented to help responsiveness as some large graphs require seconds to update a single time step. The force is from bl.ocks.org/ and documented in the code. The graph is interactive, with a click and hold able to drag a node anywhere the mouse is. A double click then returns the node to a stable position based on the force applied. This overrides the 'r' setting, turning on force.

The nodes are visualized using two attributes, helpfulness and size. The helpfulness is used for the node sizes, with more helpful being larger. The color of the nodes is determined by the average score of that reviewer.

With these techniques we can visualize how certain reviewers think alike and also how popular some reviewers are. We notice some reviewers are highly connected, thus they have a large amount of reviews. There is no outstanding result regarding helpfulness and popularity as helpfulness seems spread throughout the graph.

Fisheye and Visual Techniques

As mentioned in the last section, there are a few attributes of edges and nodes that affect the visualization of the graph. To make it user friendly, we added the ability to interact with dragging. This lets the user rearrange the graph in various layouts.

Along with force, the user is able to use fisheye to help visualize the graph. Fisheye expands the view of a circular selection around the mouse. This, like force, is also able to be toggled on and off. The user can press ‘f’ to turn fisheye on or off. By default it is turned off. This helps runtime of larger graphs and prevents any strange behavior in initialization. Another convenience of the toggle of fisheye is that the fisheye pauses wherever the mouse is when it is turned off. This can be used to zoom in on a region, but then keep that visual layout to further examine that region.

Another feature that was added was a mouseover feature, which highlights the current node that the mouse is over, as well as its edges and all the nodes it is connected to. The nodes and edges that are not associated with the node that the mouse is over are decreased in opacity to increase clarity. This mouseover feature paired with the fisheye provides a clearer visualization of the data.

Division of labor: Data scraping and analysis is done by David Campbell. Data visualization is done by Ross Donatelli.