APMA E4990 Modeling Social Data
Professor Wiggins and Hofman
Jeff Hudson, Jonathan Campbell, Xu Huang
May 8th, 2015

FINAL PROJECT

Motivating Problem

For our project we built and tested social recommendation algorithms using Yelp restaurant reviews in the state of Arizona. We had three main questions driving our research. First, can social network data be leveraged to improve the predicted ratings of a recommendation algorithm? Second, to what extent are friends more likely to have similar tastes? Third, can business-related metrics be good measures of rating and price level prediction from a visual perspective? The motivation for these questions is both commercial and general interest: companies who recommend products to users and also have access to their users' social network data, may be able to improve their recommendations, creating more value for their users. Additionally, knowing that users' friends likely share the same tastes could help better target advertising at groups of friends rather than individuals. Furthermore, knowing users' preferences help companies target specific group of users more efficiently. Finally, if people with high network centrality are found to be influential in determining ratings for their neighbors, then businesses may want to target these influential users with special promotions.

Data

The data were provided directly from Yelp as part of their Dataset Challenge[1]. The data were downloaded in JavaScript Object Notation (JSON) format, which we then converted into Comma Separated Values (CSV) files using a JSON-to-CSV converter in Python[2]. The data was

[1] To download the full dataset, complete the form at https://www.yelp.com/dataset_challenge/dataset
[2] This converter can be found at: https://github.com/Yelp/dataset-examples

extremely large (over 1.6 million reviews, 61,000 businesses, and almost 3 million social connections between users). To facilitate our analysis, we chose a subset of the data: restaurant reviews in Arizona.

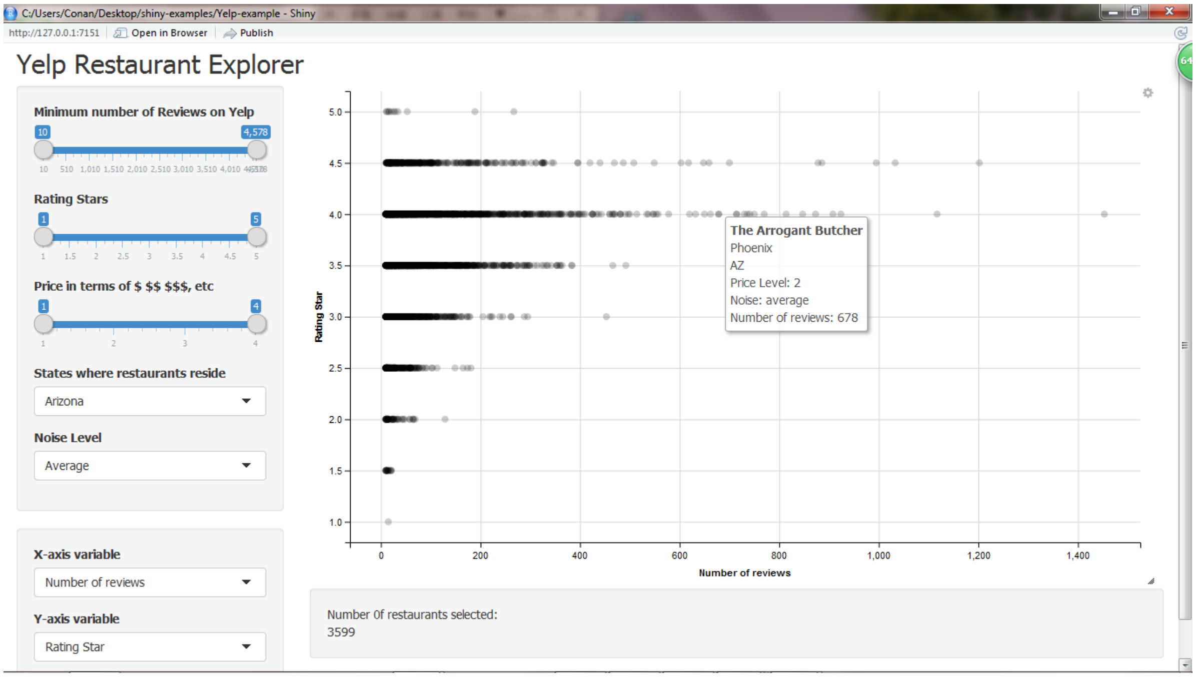
Visualization

The visualization was done in R using Shiny RStudio and Excel. The data came from the business portion of the dataset containing 61,000 businesses with 481,000 business attributes, such as name, review count, stars, noise level, and more. As aforementioned, the dataset is extremely large and the focus is on restaurants across different countries.

Once we filtered out all the restaurants from the business dataset, we decided to create a new base dataset containing only those restaurants with actual price level and with more than 10 reviews, the reason being that it is not particularly informative to study the impact of different metrics on restaurant ratings with restaurants that have fewer than 10 reviews. In reality, more than 50% of the restaurants in the dataset have fewer than 10 reviews. After removing them, we had 13,966 restaurants in our dataset as a whole.

In the beginning, we intended to segregate restaurants by cities in which they reside given that we were primarily working with large cities such as Las Vegas and Phoenix. But a closer look at the dataset revealed that there were almost 170 cities of all sizes, some of which were rather small while others were satellite cities in the Greater Las Vegas or Phoenix area. Given that each city came from one particular state, we opted instead to segregate restaurants by states, which seemed to be a better representation of the data. As a result, 170 cities were now reduced to 12 states. One thing worth mentioning is that state “EDH” and “MLN” both correspond to City of Edinburgh in the UK.

In addition, we created an interactive web application in R and Shiny to allow users to tinker with different parameters in the dataset and hopefully gain some new insights. In the application, users can choose the range of review counts, ratings in terms of stars, and price levels for the restaurants. Users can also pick the restaurant location and ambient noise level. The option to choose a number of covariates for the x and y axes also added another layer of flexibility. In general, the graphs confirm several pieces of conventional wisdom. For instance, plotting average star ratings against review counts (shown in the picture below) shows a normal distribution centered around star rating 4 with a longer tail trailing along the ratings less than 4. This pattern is expected because restaurants with better reviews tend to get more customer traffic and thereby more reviews, thus creating a positive cycle. And restaurants with consistent lower reviews get less and less traffic and eventually get weeded out, while restaurants with perfect reviews on average are very hard to come by because different reviewers might have different tastes and even a great restaurant cannot please everybody.



C:/Users/Conan/Desktop/shiny-examples/Yelp-example - Shiny

http://127.0.0.1:7151

Yelp Restaurant Explorer

Minimum number of Reviews on Yelp: 10 – 4,578

Rating Stars: 1 – 5

Price in terms of $ $$ $$$, etc: 1 – 4

States where restaurants reside: Arizona

Noise Level: Average

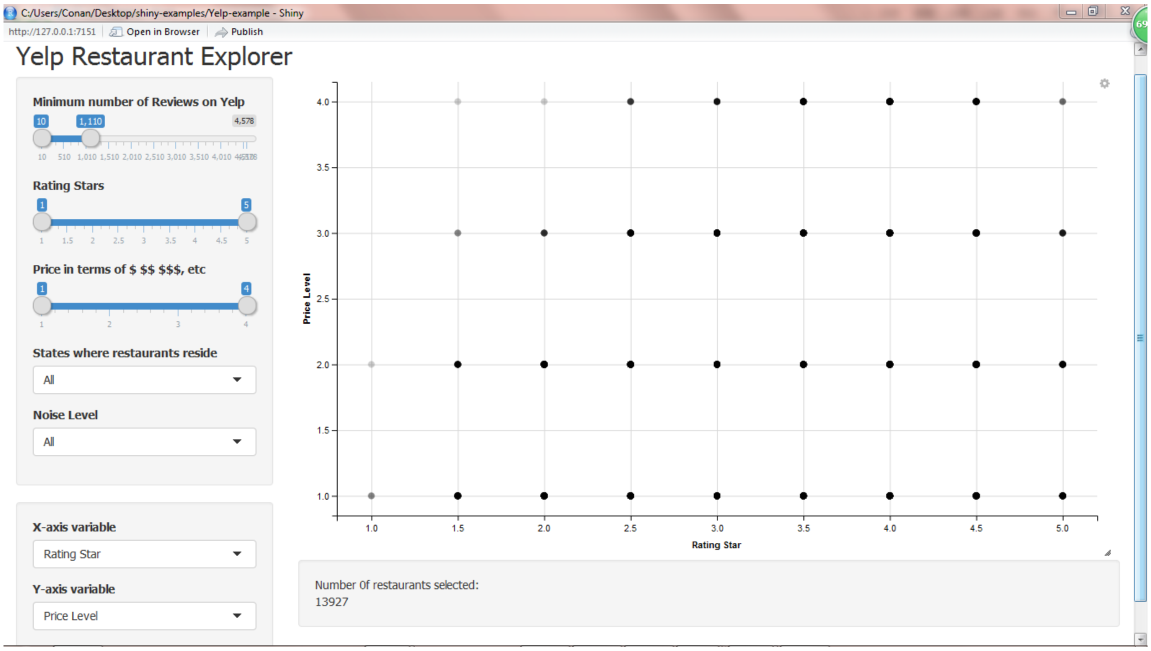
X-axis variable: Number of reviews

Y-axis variable: Rating Star

The Arrogant Butcher
Phoenix
AZ
Price Level: 2
Noise: average
Number of reviews: 678

Number Of restaurants selected:
3599

However, it goes without saying that there are several drawbacks with regard to the interactive graph. 1) Although each point representing a particular restaurant gives the corresponding average star rating and price level, there is no intuitive way to show the distribution of restaurants with different ratings given one particular price level or the distribution of restaurants from different states given the same average rating and noise level. All we can see is just lines populated by points and darker color indicates line segments in which more points are concentrated. 2) Since both average rating and price level take on a limited number of values, points belong to the same set of rating and price level are stacked on top of each other and the graph becomes uninterpretable as we select these two variables as x and y axis (as shown below). These problems are the motivations that lead up to more informative and visually-expressive graphs.



C:/Users/Conan/Desktop/shiny-examples/Yelp-example - Shiny

http://127.0.0.1:7151

Yelp Restaurant Explorer

Minimum number of Reviews on Yelp: 10 – 1,110 (max 4,578)

Rating Stars: 1 – 5

Price in terms of $ $$ $$$, etc: 1 – 4

States where restaurants reside: All

Noise Level: All

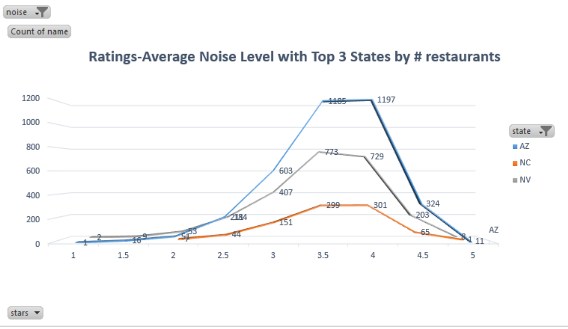
X-axis variable: Rating Star

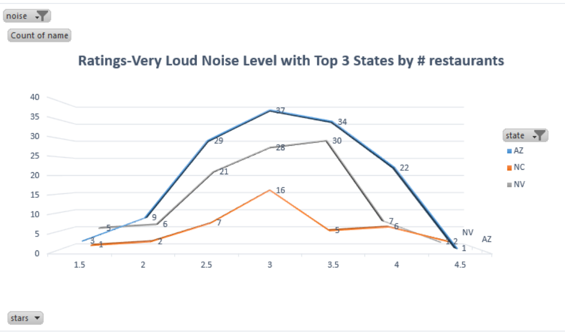
Y-axis variable: Price Level

Number 0f restaurants selected:
13927

In the light of problems we discussed about the interactive graph, we resorted to Excel Pivot Table to create graphs that contain more information. We generated 3-D graphs whose x-axis is states, y-axis is average rating, and z-axis is review count. Since Arizona, Nevada, and

North Carolina alone accounts for over 75 percent of the restaurants in our dataset, it is reasonable to use them as the x-axis. From the graphs, we saw that noise level did affect ratings across three states, holding all other variables constant. When the noise level is “average”, the majority of ratings for each state lie between 3.5 and 4. However, when the noise level increases to “very loud”, most ratings are now situated between 2.5 and 3.5. Thus we see a downward shift average ratings as noise level grows.


Ratings-Average Noise Level with Top 3 States by # restaurants

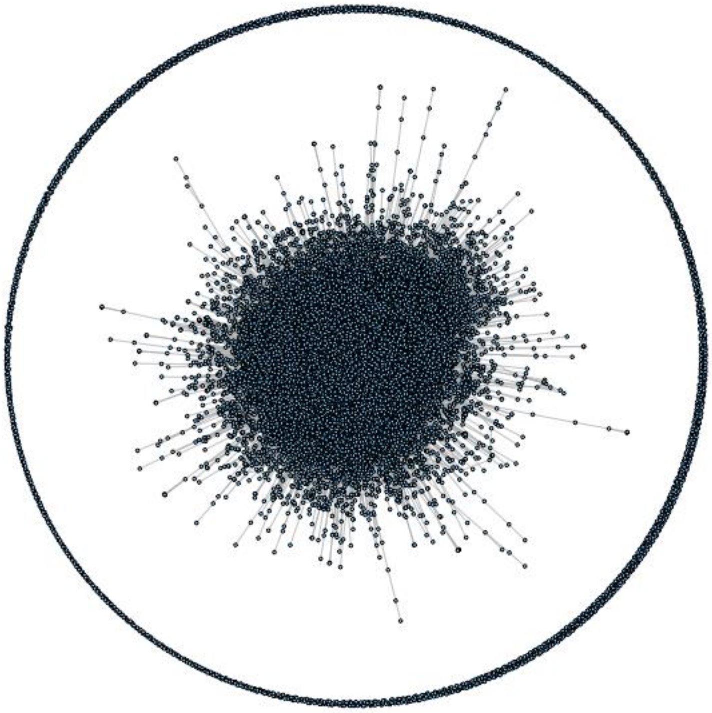

Ratings-Very Loud Noise Level with Top 3 States by # restaurants

Networks

The Yelp network provides a unique environment for a recommendation system, in that it is a social network focused on consuming the ratings of others. Not only can one ‘friend’ others, you can anonymously follow another user by becoming their ‘fan,’ making it easier to stay up to date with your network’s latest ratings. While collaborative filtering based recommendation systems have been shown to be highly predictive of user preference, we thought this would be a prime opportunity to explore the potential of network based recommendations. Due to the immense size of the network, we opted to subset on the largest component of the dataset, reviews from Arizona.

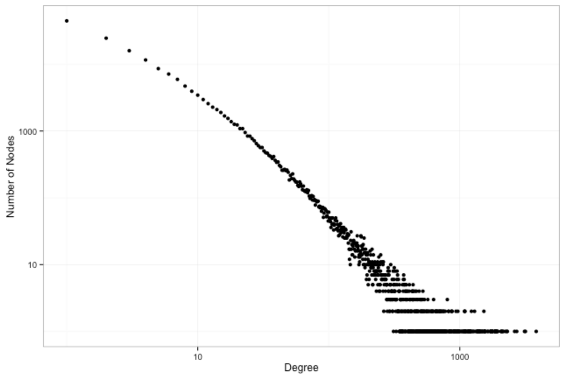
The social component of the data came to us in a wide edgelist format. Each user was characterized by a unique identifier and was accompanied by a list of all users that they are friends with. In order to examine and visualize the network, the dataset was transformed into a long undirected edge list, where each row is characterized as an edge in the network, without duplicates. Once in this format, we were able to read it into igraph to see the network. From the network structure we can not that the majority of the network is highly intertwined with a large amount of isolated friend clustered being pushed to the outskirts of the graph.

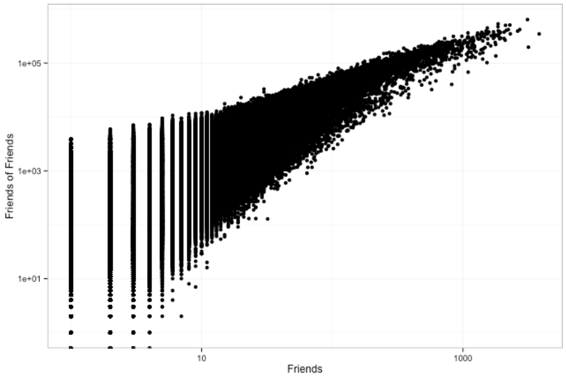
Arizona's Yelp Network



With this picture in mind, we moved forward to identify the reach of each individual's personal network. Through aggregation, we were able to use the long edgelist to calculate the degree of each individual node. Quantifying each individual's second degree reach proved slightly more difficult, given the vast size and connectivity of the network. The calculation was able to accomplished though squaring an adjacency matrix, removing first degree connection, and resetting all non-zero factors to one.

The reach of each individual gives us more insight into the relationship of the network ties. From the first graph below, we see that there are a relatively small number of actors are driving network connectivity through their vast personal network. Moving out one degree, we can see in the next graph that while one’s second degree reach is positively related to your first degree connections, the benefit decreases much more rapidly when connected to central figures in the network.





Recommenders

To establish a basis for comparison for our socially-integrated recommenders, we first needed to build garden-variety recommenders. We used one of each of the two types described in class: memory-based and collaborative filtering. To prepare the data to be used for making recommendations, we filtered the observations, keeping only the ratings from users who had rated at least ten restaurants in the state of Arizona. We then split the ratings data, holding out 10% as a test set. Any businesses that were present in the test set but not in the training set, were removed from the test set.

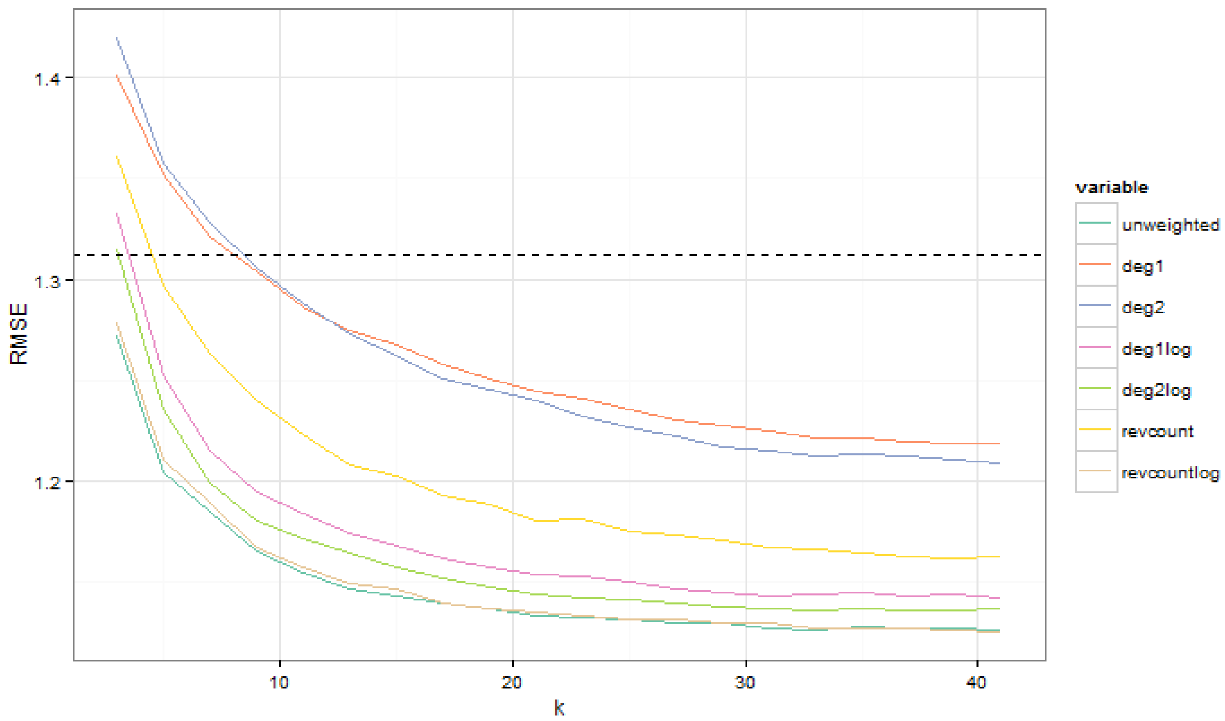
First we used probabilistic matrix factorization to implement the collaborative filtering recommender. In this recommendation system, we used the same 10% split in the data. During the training stage, the algorithm attempts to represent the full user-by-restaurant rating matrix as a product of two smaller matrices of rank k. It does this by coordinate ascent, solving miniature ridge regression problems for each user as a product of the restaurant features, then for each restaurant as a product of user features. Because this algorithm requires many iterations, each of which took about a half hour (given the size of the dataset) we did not have the opportunity to assess the effect of model complexity (rank of the matrices) on the accuracy of the predictions. We used k=20 because that is a common choice for collaborative filtering problems. After 50 iterations the collaborative filtering algorithm had a root mean squared error of 1.312.

The matrix factorization problem was very computationally intensive. It took many hours to run on a single machine, only processing a subset of the total dataset. At real-world scale, such an algorithm would need to be run offline in a distributed framework such as MapReduce. Though we did not implement this for our project, the Mappers would solve the small ridge regression problems for each user or restaurant, then the Reduce step would collect the results and update the matrices. Because it is an iterative algorithm, it would take many Map/Reduce cycles to run fully, but this should still give an improved running time compared to a single machine.

In addition to collaborative filtering, we also used the k-nearest-neighbors, memory-based recommendation algorithm. The kNN recommender uses Euclidean distance to compute similarity between users (based on their ratings in the training set) and returns the average (mean) rating of the k nearest. To test the accuracy of these recommenders, we again used root mean squared error (RMSE) on the test set. For the kNN recommenders, we plotted this measure as a function of k to determine how the model's accuracy changes with increasing complexity. For all models it improves the accuracy quite a bit at first, then the marginal improvements shrink till it plateaus.

In an attempt to improve on the kNN recommender, we used several different user attributes to weight the recommendations from each neighbor. We used the node degree (how many friends a user has in the network) and the second degree (friends-of-friends), both as raw counts, and logged counts. We also included review count (raw and logged), to see if recommendations could be improved by upweighting more prolific reviewers.

Across all three weight measures, the logged values performed better than the raw counts (as measured by a smaller RMSE). Unfortunately, none are an improvement on the generic kNN recommendation algorithm. The recommender weighted by the log of review count comes very close, but ultimately, since the RMSEs are indistinguishable, we prefer to go with the simpler model. The following figure shows the RMSE of each recommender as a function of the model's complexity (the value of k). The dashed line indicates the RMSE of the collaborative filtering recommender after 50 iterations.



Conclusion

Our preliminary analysis found that including social network data did not increase the accuracy of the predictor. Further analysis might try including more sophisticated measures (such as centrality) or more complex ways of integrating them into the recommendation (rather than simply as weights to kNN). Additionally, we could use the social networks themselves to motivate the predictions. This approach would be similar to kNN but would use distance defined by the social network itself.