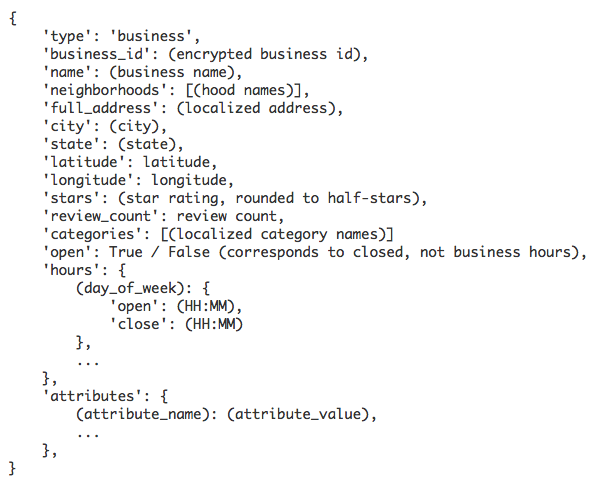
Samantha Merrill CSC 240

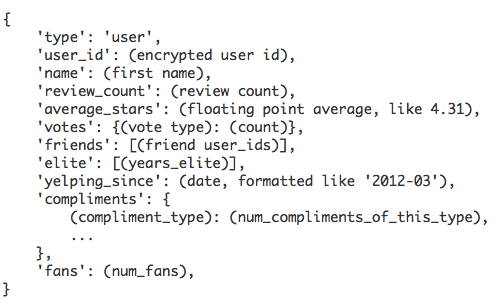
Fall 2014 Final project

My project focuses on Yelp recommendations. Surprisingly, Yelp currently does not do much in the way of recommending businesses to its users. It shows “recommended reviews”, which are reviews users have posted that Yelp has deemed most helpful or the highest quality. Yelp also offers advertising space businesses can purchase. It seems to be an open question how Yelp can use its data to generate recommendations for businesses users may like but have never reviewed or visited. A traditional recommendation model suggests businesses that are the same type as the one a user reviewed. For example, if a user reviews a restaurant then they would get a recommendation for another restaurant. In my project I wanted to do something different and instead make recommendations for different types of businesses from the one a user visited. If a user reviews a restaurant, I will recommend another type of business such as a movie theater or clothing store.

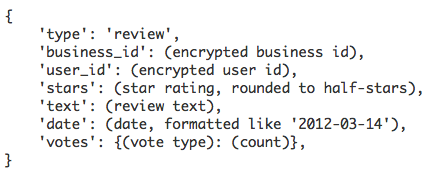
Yelp has provided data in the json format. The relevant data files to my project are businesses, users, and reviews. The businesses span the cities of Phoenix, Las Vegas, Madison, Waterloo and Edinburgh and the data includes the following attributes:



The users data had the following attributes:



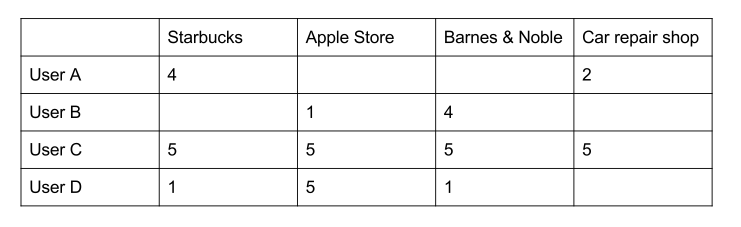
And finally the reviews data had these attributes:



The hardest part was figuring out how to generate recommendations in an efficient manner. One idea I had was to use the apriori method to first find users that were similar to the user in question by finding who had rated the same places highly. Then I tried to find out which other establishments frequently appeared with that one. Unfortunately this approach did not work because it did not finish running. I was not surprised because we talked about this problem with apriori in class. However, I had to find a better way to generate recommendations that took a reasonable amount of time.

I researched recommender systems and found a book chapter by Jure Leskovec et al. that laid out a nice way to create an efficient system. I modeled my project based on this approach. There are two types of recommender systems: content-based and collaborative filtering. Content-based finds items that have similar characteristics to items a user rated highly. Collaborative filtering finds similar users and recommends items that they liked. I chose to use collaborative filtering because it is conducive to recommending different types of businesses. It would be ineffective to use content-based recommending for my purpose because it would only recommend businesses that are similar to the one you already rated.

Using collaborative filtering relies on creating what is called a utility matrix, which is a table consisting of every user and business and the user’s rating of the business if it exists. A utility matrix looks like the following:



The goal is to infer the blank ratings. The way to do this is to find similar users and use the average of their ratings for a given business. How do we find similar users? There are different distance measures we can consider including Jaccard index and cosine similarity. The Jaccard index is problematic in this case because it considers blank ratings to be similar to each other. We only want to take into account the ratings that do exist because if two users have not rated a business, they might end up having opposite opinions of that business. Cosine similarity could work better because it measures the angle between two user rating vectors. One question I had was whether to normalize the data. I would do this by subtracting the user’s average rating from each rating they gave. This would have the effect that the businesses they liked would have positive rating numbers and the ones they did not like would have negative ratings. Normalizing the data would cause the variance in angles to increase because sometimes the vectors would point in opposite directions.

Another decision I had to make was whether to find recommendations for one user at a time or to find them for all users at the same time. If I were to just find recommendations for one user I could simply find the most similar users to that one user and then see what they rated highly. If I wanted to find recommendations for all users I could use a clustering algorithm to cluster them into groups of similar users. For this project I decided to use the one-at-a-time approach because Professor Luo recommended it and I wanted to have more control than using a pre-made clustering algorithm.

I coded my project in python and it does the following steps:

1. Read in json data files for users and reviews
2. Use them to make a dataset. The dataset is a dictionary with user\_id as the key and another dictionary as the value. The value dictionary contains a list of the user’s reviews – just the number of stars and the business\_id – and optionally, some profile information. I didn’t utilize the space for profile info because Yelp does not provide much helpful information about users. Helpful information would include age, gender, etc.
3. From the dataset, make a utility matrix. It is a 2-dimensional array (list of lists in pythonic terms) where the indices are unique identifiers for users and businesses that I store in another dictionary.
4. Optionally, normalize the matrix by subtracting each value from the user’s mean.
5. Find similar users. This step utilizes a cosine similarity function I wrote. It probably takes the longest of all the steps because performing this task for all users would result in O(n^2) runtime. I allow for a threshold parameter that only returns users that have a similarity above a certain threshold. I set the threshold to 0.3 using the normalized utility matrix because any higher and there aren’t enough similar users to make recommendations.
6. Make recommendations. This is done by taking the similar users found in the previous step and returning the businesses they rated highly. I rely on a function I wrote that returns a list of favorite businesses for a given user. I deem it a favorite if its rating is > 0 because that means the user rated it above average.

Bibliography

Anand Rajaraman and Jeffrey David Ullman. 2011. Mining of Massive Datasets. Cambridge University Press, New York, NY, USA.