COMP 4601 - Assignment 2

Student Name: Brian Ferch
Student Number: XXXXXXXXXX

Student Name: Jules Kuehn
Student Number: 100661464

Winter 2019

1 Generating user profiles from reviews

A wealth of metadata was contained within the /reviews/ HTML files. This included:

- Movie ID (ex. B0000D0XZ4)

- User ID (ex. A1A69DJ2KPU4CH)

- User profile name (ex. James Ferguson)

- Rating of movie (ex. 5.0)

- Review helpfulness (ex. 64/74)

- Review time (ex. 1287273600)

We scraped all of this data except the review times, although this data has value as well (see "Future work" below). For the purposes of this assignment, the simplification / assumption is that users have static preferences over time.

While it is certainly possible to do sentiment analysis on the individual reviews (as we have done for the in-class assignment), we felt that given these "true" user ratings, it made more sense to make use of this explicit data than any extracted sentiment. So, the reviews data was processed into the following format:

- A "ratings table" where each row label is a User ID, and each column label is a Movie ID. The data contained in this table is the rating of the movie. The rating scale is from 1 to 5, but we stored the values as between 0 and 1, with a value of -1 for a movie not reviewed by this user. The formula for converting star ratings to the range 0 to 1 is $rFloat = \frac{(rStar-1)}{4}$.

- A "helpfulness table" corresponding with the same dimensions and labels as above, but storing the helpfulness of each review (parsed to a float)

- A map from User ID to User profile name

Given this information, some useful characterizations can be made on each user. Included in each of our user profiles is that user's:

- Average star rating

- Average helpfulness (as rated by other users)

- Community affiliation

- Position in relation to other users in 2d space

2 Clustering users into communities

The ratings table is large (1252 rows x 1079 columns) and very sparse (93.9%). We first reduced this to a dense, low-dimensional representation by the following process:

1. Calculate average ratings for each movie, each user, and the overall average

2. Fill the missing ratings with the movie's average rating, adjusted to the user's average rating:
   $rFill = movieAvg + userAvg - overallAvg$

3. Normalize the matrix to adjust for user average ratings in general.

4. Determine an appropriate number of dimensions k. We can see below that the user clusters determined from the 200-dimensional representation are very similar to the 2-dimensional representation, and that the vast majority of the variance occurs in a single dimension.
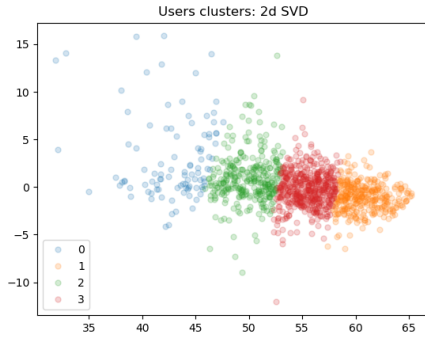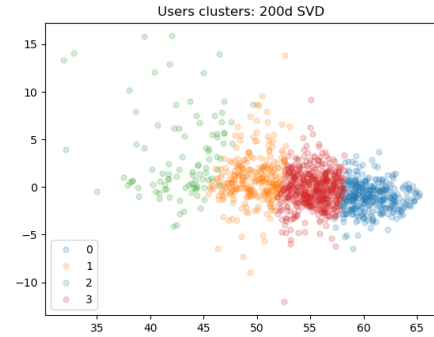

Figure 1: Kmeans clustering on users in 2d

Figure 2: Very similar results in 200d

Given these results, we used a 2-dimensional representation for users. We determined the best number of clusters to be $m = 4$ based on elbow finding in both 2d and 200d. The process ultimately generated two useful mappings: User ID to 2D representation, and User ID to community number.

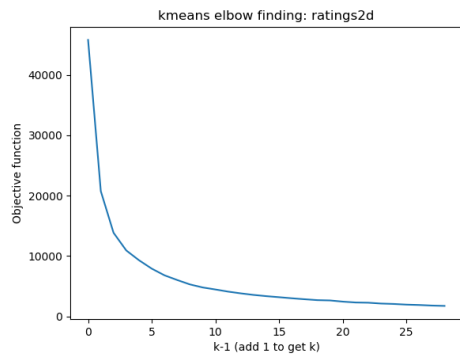## 3 Clustering movies according to page text

...

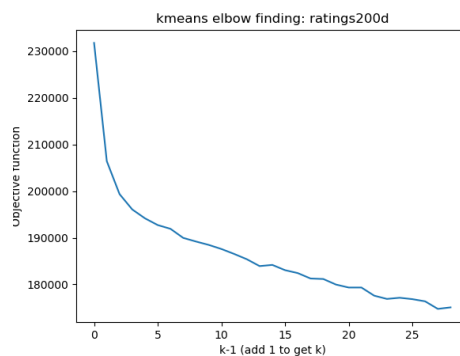Figure 3: Elbow finding in 2d suggests 4 clusters

Figure 4: Similar results in 200d

4 Generating and retrieving contextual advertisements

...

5 SUGGEST algorithm

...