COMP 4601 - Assignment 2

Student Name: Brian Ferch
Student Number: 100962115

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 (users) x 1079 columns (movies). It is also very sparse (93.9% empty). 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 for truncation.

Reconstructing the original matrix from the 2-dimensional truncated SVD yields an error of 29.2% without filling or normalization. After filling and normalization though, the error is reduced to 23.7%. Truncating at 200 dimensions, this error drops to 9.3%. However, this has little effect on the clustering of users. 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.

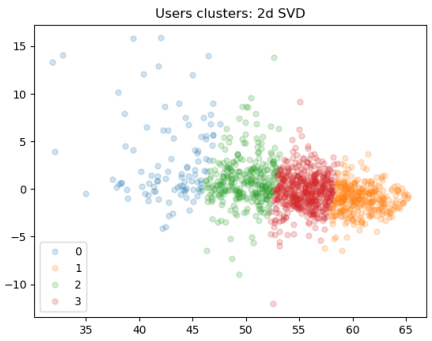

Figure 1: Kmeans clustering on users in 2d

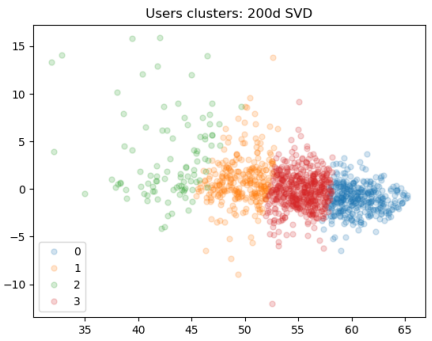

Figure 2: Very similar results in 200d

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.

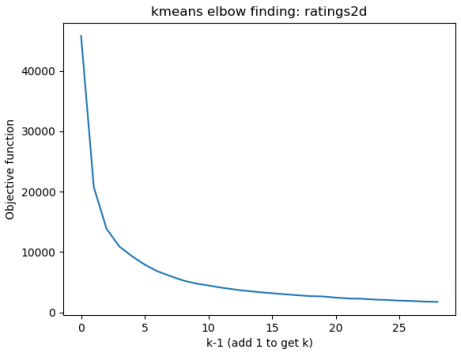

Figure 3: Suggests 4 clusters of users

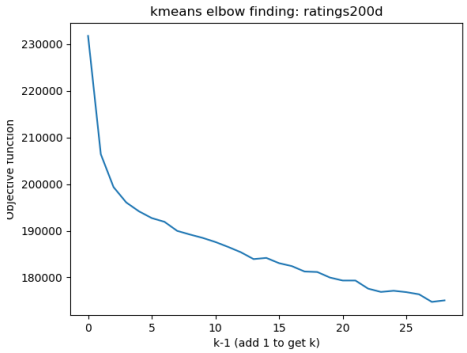

Figure 4: Similar results in 200d

3 Topic analysis on movies according to page text

We could create some approximation of genre for each movie by extracting topics from the pages with the following process:

1. Calculate LDA topics on review pages (since each page corresponds to one movie).
2. For each page, we take the topic with the maximum distribution value and call this the movie's topic.

Tinkering with the parameters of LDA iteratively led us to topics that progressively came closer to mimicking genre.

We strived to keep our distributions fairly biased to one topic since only the max was considered. Four topics ended up being the sweet spot that gave us this behavior.

Horror	Evil
Original	Classic
People	Something
Dead	Character
Michael	Night
Story	Pretty
Little	Part
Remake	Actually
New	Things
Living	Young

Figure 5: Example "horror" topic

Story	New
Man	Performance
Character	Real
Life	War
People	End
Action	Another
Role	Director
Little	Something
Last	Bad
Bourne	John

Figure 6: Example "war action" topic

Stop words were added at each iteration as they appeared in the generated topics. In addition to conventional stop words, topic words that were too general and not indicative of genre like "movie"

or "film" were added.

4 Generating and retrieving contextual advertisements

Having now generated $m = 4$ user communities and $n = 4$ movie topic categorizations, we create $m * n = 16$ advertising categories, one for each combination of {community, topic}.

Advertising for each community is comprised of "recommended movies" for that community. These movies are determined by:

- Creating a community ratings matrix where the ratings are the averages of all ratings from users in that community for a particular movie (or -1 if no users in this community rated the movie)
- Finding the movies for each community which are rated better (by that community) than the community average

An advertisement is generated for a specific {user/page} by:

- Retrieving all recommended movies for the user's community
- Shuffling this list so as to provide sampling from the long tail, rather than just the best rated movie
- Taking the first movie from the shuffled list that matches the topic of the current page (or a random recommended movie if the community has not recommended any for this topic)
- Generating some HTML which includes the recommended movie's star rating and a link to reviews

5 SUGGEST algorithm

We are given a social graph connecting a new user (with no reviews) to other users, who we are assuming have reviews. The goal of our suggest algorithm is to position the new user in the 2d ratings space based on users who are close in the social graph. We can then associate the new user with a particular community (the closest cluster center to this user in 2d).

A directly connected user is said to have $d = 0$ while a user with one degree of separation on the social graph has $d = 1$ and so on. The maximal degree we consider can be set by experimentation. Users with closer degree have increased weight in the calculation of the new user's position:

$$p_{new} = \frac{\sum_{i \in U} \frac{p_i}{d_i+1}}{\sum_{i=1}^{k} \frac{1}{d_i+1}}$$

p_i : position of user with ID i in 2d ratings space
U : set of IDs for k nearest neighbours of new user

The movies that we recommend for the new user are the "recommended movies" (described in previous section) for their computed community. The same system can be used for advertising as before.

We could also modify our advertising system to take advantage of the social graph wholesale. Instead of creating ratings communities, we take the users with close social proximity, and calculate "recommended movies" for this neighbourhood. Recommended movies for a user's neighbourhood are those predominantly rated greater than or equal to the user average by a user's k nearest neighbours in the social graph.

A combination of the two approaches - closeness in rating behaviour and closeness in social graph - may provide the best results. Once a new user add their first rating, their 2d ratings representation should be updated right away. In the short term, we will use previously calculated (and now out-of-date) movie and user averages to fill and normalize the new user's ratings and fold this data into the existing SVD. We would make a full rebuild of the truncated SVD model periodically.

6 Improvements

For this assignment we made the assumption that users have static preferences over time. In reality, this is not a good model for profiling users. Since the metadata of the reviews include timestamps, we could profile the users based only on recent behaviour - or even predict the trajectory of user preferences. Taking a sliding window of user behaviour over time would yield a different (2d) representation for the same user at different time stamps. From this, we can use regression to predict the 2d representation of the user at some point in the future, and use this representation for generating advertising.

Upon inspection of the user ratings, we found that they were strictly integers from the set {1, 2, 3, 4, 5}. Instead of filling the sparse matrix with item averages (which, in such a sparse matrix, can "smear" the useful information towards the mean) we could use a one-hot encoding on the integer ratings.

Different techniques could be tried for creating lower-dimensional representations of users, including PCA and t-SNE. This could improve clustering.

The helpfulness of a review, or the average helpfulness rating for a user, could be used to give increased weight to high quality reviews.

Improvements can be made to the cleaning and parsing of the HTML. We noticed that there were duplicate reviews in the dataset corresponding to duplicate pages (movies which have the same reviews, but different IDs). These should ultimately be removed from the dataset before processing. The reviews HTML also included the title of each review, which should be included in the text used for LDA topic categorization.

Experimentation with different stopwords may also lead to better results in the LDA process.

Another approach to the categorization process begins with creating document vectors for each of the pages (a concatenation of movie reviews, ideally augmented with the review titles). In the same format, we would create document vectors for reviews of movies for which we know the category, and train a neural net on this latter set. Our pages would then be predicted to a known category, which would likely provide a more meaningful categorization than the unsupervised bag-of-words approach we took for the assignment.