CS 422: Data Mining

Department of Computer Science
Illinois Institute of Technology
Vijay K. Gurbani, Ph.D.

Spring 2019: Homework 4 (10 points)

Due date: Sunday, May 5 2019, 11:59:59 PM Chicago Time

Please read all of the parts of the homework carefully before attempting any question. If you detect any ambiguities in the instructions, please let me know right away instead of waiting until after the homework has been graded.

This is the final version of the homework (Apr 23, 2019).

1 Exercises (4 Points divided evenly each question) Please submit a PDF file containing answers to these questions. Any other file format will lead to a loss of 0.5 point. Non-PDF files that cannot be opened by the TAs will lead to a loss of 2 points.

1.1 Leskovec, Ch. 3, Finding Similar Items (Page numbers correspond to the online version of the book; see syllabus for the URL.)

Page 77, Exercise 3.1.1; Page 80, Exercise 3.2.1; Page 86, Exercise 3.3.3; Page 91, Exercise 3.4.1 (evaluate the S curve only for the first two bullets); Page 97, Exercise 3.5.4; Page 98, Exercise 3.5.5.

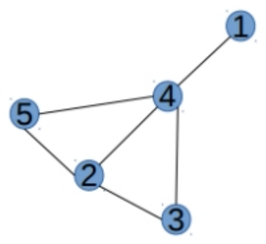
1.2 Tan, Ch. 2, Questions 14, 18, 19, 20.

1.3 Leskovec, Ch. 5, Link Analysis (Page numbers correspond to the online version of the book; see syllabus for the URL.)

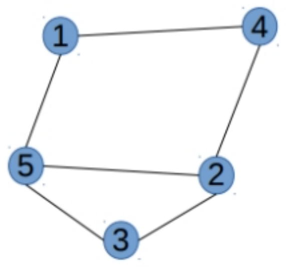
Page 175, exercise 5.1.1; page 176, exercise 5.1.2; page 177, exercise 5.1.6

1.4 Centrality Measures

Consider the following two graphs:



(a)



(b)

For each graph, calculate (SHOW ALL WORK):

(a) Normalized degree centrality for each node (b) Normalized closeness centrality of each node

(c) Normalized betweenness centrality of each node

2 Practicum problems (6 points divided evenly by each assignment) **Please label your answers clearly, see Homework 0 R notebook for an example (Homework 0 R notebook is available in "Blackboard→Assignment and Projects → Homework 0".) Each answer must be preceded by the R markdown as shown in the Homework 0 R notebook (### Part 2.1-A-ii, for example). Failure to clearly label the answers in the submitted R notebook will lead to a loss of 2 points per problem below.**

2.1 Topic: Locality sensitive hashing (2 Points)

In this problem, you will run LSH or Locality Sensitive Hashing. Make sure you set the seed to be **100** when you invoke textreuse::minhash_generator(). **If you do not do this, your output will not match expectation and points will be taken off!**

You are provided a corpus of 100 documents (see corpus.zip). This corpus is donated by Paul Clough and Mark Stevenson (University of Sheffield). They designed this corpus to represent varying degrees of plagiarism and hoped that it will be a useful addition to the set of resources available for the evaluation of plagiarism detection systems.

Read this corpus in by unzipping the ZIP file in a directory. Let's assume you unzip it in /home/vkg/corpus. To read all the files in, use the following R command:

```
file s <- list.files("/home/vkg/corpus", full.names=T)
```

The above command will cause all of the files to be stored in the list files, using the complete path name of the files. You can subsequently pass the files list to textreuse::TexReuseCorpus() function.

Based on this corpus, please answer the following questions.

(a) How many shingles (or tokens) are there in all of the 100 documents? (Hint: look at package TextReuse::tokens()).

(b) What are the dimensions of the *characteristic matrix*?.

(c) Print the first 5 shingles (or tokens) of the file orig_taske.txt.

(d) We will fix our signatures (or hashes, or the rows in the *signature matrix*) at 240. This represents what percentage reduction in the size of the problem?

(e) At 240 signatures (or hashes), how many bands would we need to detect a minimum Jaccard similarity of 0.23? (Hint: Use lsh_threshold().)

(f) Using the number of bands you determined in (e), what is our probability of catching similar documents at a minimum Jaccard similarity of 0.23?

(g) We will first use the brute-force method to determine how many documents are similar. To do this, run pairwise_candidates() on the characteristic matrix (i.e., original data in the corpus).
(i) How many comparisons were made when we used the characteristic matrix?
(ii) Examine the object returned by pairwise_candidate(). The object returned is called a tibble. Tibbles are data frames except that they are invariant, i.e., once created, they do not change. From this object, how many documents have a Jaccard similarity score of at least 0.23?
(iii) List all the rows in the tibble that contain a Jaccard similarity of at least 0.23, sorted in decreasing order by the score.

(h) We will now use LSH to determine how many documents are similar. To do this, use lsh() and lsh_compare() find the candidate pairs using the bands determined in (e).
(i) While running LSH on the corpus, how many comparisons were made?
(ii) Compared to the number of comparisons made in (g)(i), how much was the percentage *decrease* in the computation needed to find the candidate pairs?
(iii) List all the rows in the tibble that was obtained in (h), sorted in decreasing order by the score. How many rows are there in the tibble?
(iv) How many rows contain a similarity index of at least 0.23?
(v) List all the rows in the tibble that contain a similarity of at least 0.23, sorted in decreasing order by the score.

(i) Examine the output of g(iii) and h(v). Comment on the output in terms of how well LSH did in catching duplicate documents.

2.2 Topic: Content-based recommendation system (2 Points)

In this assignment, you will develop a small content-based recommendation system. The MovieLens dataset (http://movielens.org, a movie recommendation service) is curated by a research group at the University of Minnesota (https://grouplens.org/datasets/movielens/). You will be using the *small* dataset that consists of 100004 ratings and 1296 tag applications across 9125 movies. These data were created by 671 users between January 09, 1995 and October 16, 2016. This dataset was generated on October 17, 2016. This dataset is available to you from the Blackboard homework 4 site (see ml-latest-small.zip file).

For this problem, you will focus on two files in the dataset: ratings.csv and movies.csv.

The file ratings.csv contains ratings of 671 users, each user is identified by a unique user ID. The file movies.csv contains movies, titles and genres, each movie is identified by a unique movie ID.

Your task is to develop a movie recommender engine that will create a profile of a user and then match 10 randomly selected movies to the user's profile; the top 5 movies will be suggested to the user as possible movies to watch.

Each student will select a user to profile as follows: Take your Illinois Tech ID (minus the 'A' prefix) and perform modulo division by 671 (the number of users in the small movielens dataset). The answer to the division is the user ID whose profile you will construct. So, if for instance, your Illinois Tech ID is A55556666, then 55556666 mod 671 = 550. Thus, you will use user ID 550 and build a profile for that user. See below how to build the user profile.

Once you have constructed the user's profile, you will randomly choose 10 movies from the movies.csv file. As an example, let's say you randomly chose the following 10 movie IDs: 1967, 7319, 4602, 1550, 3355, 91535, 4297, 4169, 606, 1361. Then, the movie with movie ID 1967 in the movies.csv file is "Labyrinth (1986)"; the movie with movie ID 7319 is "Club Dread (2004)"; the movie with movie ID 4602 is "Harlem Nights (1989)", and so on. You will create a profile for each movie. See below how to build the movie profiles.

Once you have constructed the user's profile, and 10 movie profiles, you will use the cosine similarity metric to get the similarity between the user and each of the movies. You will then output the top 5 movies with the highest similarity that match the user's profile. R has a package called lsa from which you can get the cosine similarity function. Alternatively you can write your own, or you can borrow the code for the cosine similarity function I provide on slide 15 of the Recommender Systems Part 1 lecture (Lecture 12).

Building a user profile

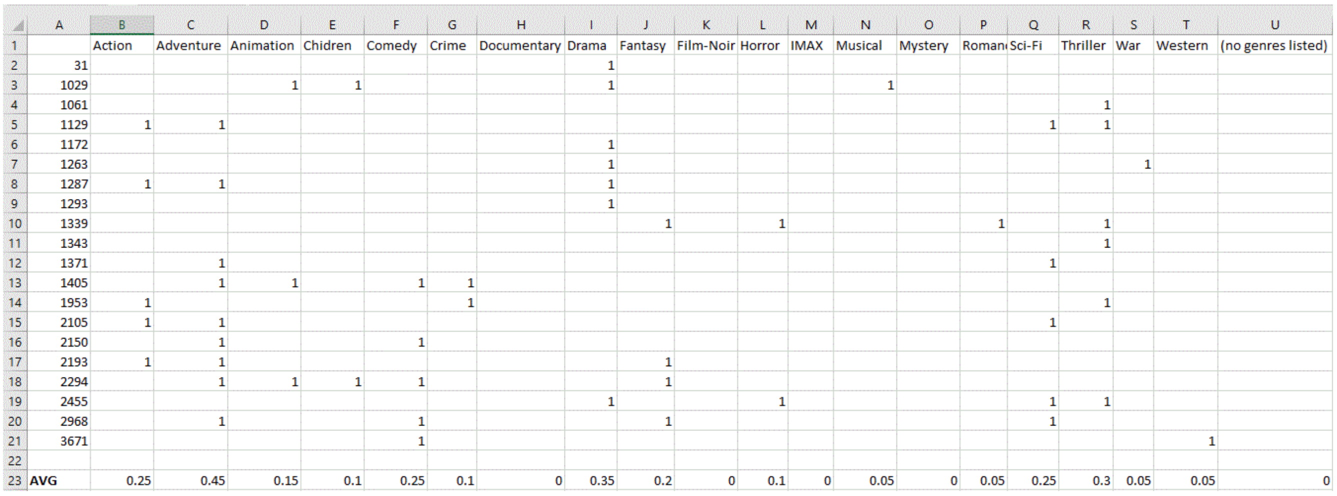
To build a user's profile, you will examine all of the movies that the user has watched. (Be careful, one user has actually watched 2,391 movies and if you happen to randomly pick that user then make sure you test your code on a user who has watched less number of movies. In fact, it is a good idea while developing the user's profile that you do it on a user ID that has watched the least amount of movies. The least amount of movies watched is 20, and user ID 1 is a good example of such a user.)

The profile will be constructed in 20 dimensions. Each genre of the movies that the user has watched becomes an attribute. You will go through all of the movies that a user has watched and extract the genres of the movies. The genre is a "|"-separated list. In R, you can use the strsplit() function to tokenize a string using a delimiter ("|") into its constituent tokens. Read up on the strsplit() function for more information.

These are the 20 genres that are present in the movies.csv file:

"Action", "Adventure", "Animation", "Children", "Comedy", "Crime", "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "IMAX", "Musical", "Mystery", "Romance", "Sci-Fi", "Thriller", "War", "Western", "(no genres listed)".

To build a user profile, you will create a data frame with 20 columns and as many rows as the number of movies that the user has watched. Each movie is decomposed into its constituent genres and the appropriate cells in the dataframe corresponding to the genre are set to '1'. For example, the spreadsheet below shows the decomposition of the 20 movies that user ID 1 has watched into its constituent genres (the first column is the movie ID).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Action	Adventure	Animation	Chidren	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	IMAX	Musical	Mystery	Roman	Sci-Fi	Thriller	War	Western	(no genres listed)
2	31								1												
3	1029			1	1				1					1							
4	1061																	1			
5	1129	1	1														1	1			
6	1172								1												
7	1263								1										1		
8	1287	1	1						1												
9	1293								1												
10	1339									1		1				1		1			
11	1343																	1			
12	1371		1														1				
13	1405		1	1		1	1														
14	1953	1					1											1			
15	2105	1	1														1				
16	2150		1			1															
17	2193	1	1							1											
18	2294		1	1	1	1				1											
19	2455								1			1					1	1			
20	2968		1			1				1							1				
21	3671					1														1	
22																					
23	AVG	0.25	0.45	0.15	0.1	0.25	0.1	0	0.35	0.2	0	0.1	0	0.05	0	0.05	0.25	0.3	0.05	0.05	0

To create a user profile, you will sum up the 1's in each column (or attribute) and divide them by the number of movies watched by that user. In the spreadsheet above, row 23 (labeled "AVG") is the mean of the columns. This row becomes the user profile vector:

<0.25, 0.45, 0.15, 0.10, 0.25, 0.10, 0.00, 0.35, 0.20, 0.00, 0.10, 0.00, 0.05, 0.00, 0.05, 0.25, 0.30, 0.05, 0.05, 0.00>

In the vector above, the first element corresponds to genre "Action", the second "Adventure", and so on.

Building a movie profile

To build a movie profile, you will again extract all of the genres that the movie can be decomposed into and the column corresponding to the genre of the movie gets set to a ‘1’ and all other columns get set to ‘0’. For example, movie ID number 145 is decomposed into the following genres: “Action”, “Comedy”, “Crime”. “Drama”, “Thriller”. The movie vector corresponding to this movie will be:
<1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0>

In the vector above, the first element corresponds to genre “Action”, the second “Adventure”, and so on.

Once you have built the user profile and movie profiles, you are now ready to run the cosine similarity and present the results. Again, as an example, the cosine similarity of the user profile vector and the movie profile vector shown above is: 0.666

Your output should be as follows:

```
User ID X chose the following 10 movies: 18, 47, 255, 269,  471, 445, 640, 680, 1589, 1562
Of these, the following 5 movies are recommended:
MovieId         MovieName                                    Similarity
1562            Title 1                                      0.671
445              Title 2                                     0.584
680              Title 3                                     0.221
471              Title 4                                     0.195
255              Title 3                                     0.108
```

2.3 Topic: Collaborative Filtering (2 points)

In this assignment, you will develop a small collaborative filtering-based recommendation system. We will use the same dataset that was used in Problem 2.2. Your task is to develop recommendation engine that will predict the **rating** that a user will give to a movie.

To do this, we will work with user ID 191. User ID 191 has watched (and rated) the following 29 movies:

10 34 47 110 150 153 161 165 185 208 231 292 296 300 318 339 344 349 356 380 434 454 457 480 588 590 592 593 595

Problem 2.1 informs us that there are 36 users who match user ID 191; the following 12 users with high similarity scores will be considered for this problem:

User IDs	**Jaccard Similarity**
User ID 513	0.4358974
User ID 317	0.4033613
User ID 415	0.3255814
User ID 375	0.3049645
User ID 64	0.2753623
User ID 556	0.2727273
User ID 82	0.2527473
User ID 225	0.2420382
User ID 657	0.2262774

User ID 266	0.2216216
User ID 568	0.2105263
User ID 50	0.2009804

Because we are interested in measuring the performance of our recommender engine's prediction, we will treat 4 movie IDs from User ID 191 as test observations. These four movie IDs are: 150, 296, 380, and 590. Write down the recommendation that user ID 191 provides to these movies separately and set the ratings of these movies to NA for User ID 191. We will assume that User ID 191 has not rated these movies, allowing us to predict the rating that User ID 191 would give to that movie and to check the error of our model using RMSE. Use the following formula to measure RMSE:

$$\sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

where $n = 4$. (These are the 4 movies whose ratings we are predicting.)

(a) Prediction using user-user similarity: You will randomly pick 5 users from the above table. Using these 5 users and User ID 191, you will create a utility matrix, *U*, where the rows are the **users** and the columns are the **movies**. *U* will be filled with the ratings that a particular user gave a movie, if the user watched that movie. If the user did not watch the movie, the cell will contain an NA. Each User ID in the above table may have watched more movies than User ID 191 has, or less movies, or equal amount of movies. To ensure *U* has the same number of columns, you will use the intersection of the movies that User ID 191 has watched with each of the randomly chosen 5 users. *U* will end up being a 6x29 matrix. (Hint: See ?intersect() in R to get the intersection of movies User ID 191 has with other users.)

Using the utility matrix *U*, run the collaborative filtering algorithm as outlined in the lecture on recommender systems (Lecture 13, slide 28, Option 2). The similarity you will use between users is the Jaccard Similarity given to you in the above table. You will establish a *neighbourhood* of 3 users from the 5 randomly chosen users; so |N| = 3. Use neighbours who exhibit the highest Jaccard similarities; i.e. order the 5 randomly chosen users by their Jaccard Similarity score and pick the three users with the highest scores.

Using this neighbourhood, you will predict the ratings that User ID 191 will give to these movies using the equation labeled "Option 2" in Lecture 13, slide 28.

Once you have predicted the ratings that User ID 191 will give to the 4 movies, you will calculate the RMSE.

Your output should be of the following format:

```
User ID 191, 5 random user IDs: 375, 657, 513, 50, 225.
Using user-user similarity, User ID 191 will rate the movies as follows:
150: X.X
296: X.X
380: X.X
590: X.X
RMSE: Y.Y
```

(b) Prediction using item-item similarity: You will randomly pick 5 users from the above table. Using these 5 users and User ID 191, you will create a utility matrix, *U*, where the rows are the **movies** and the columns are the

users. *U* will be filled with the ratings that a particular user gave a movie, if the user watched that movie. If the user did not watch the movie, the cell will contain an NA. Each User ID in the above table may have watched more movies than User ID 191 has, or less movies, or equal amount of movies. To ensure *U* has the same number of rows, you will use the intersection of the movies that User ID 191 has watched with each of the randomly chosen 5 users. *U* will end up being a 29x6 matrix. (Hint: See ?intersect() in R to get the intersection of movies User ID 191 has with other users.)

Using the utility matrix *U*, run the collaborative filtering algorithm as outlined in the lecture on recommender systems (Lecture 13, slides 29-33). The similarity you will use between users is the Pearson correlation similarity as shown in the slides. You will establish a *neighbourhood* of 3 items, |N| = 3. (Hint: To get the mean of each row in *U*, use apply(*U*, 1, function(x) mean(x, na.rm=T)). See ?apply().)

Using this neighbourhood, you will predict the ratings that User ID 191 will give to these movies as shown in Lecture 13, slides 29-33. (If you do encounter a case where the highest similarity row contains an item (movie) that the user 191 has not rated, simply treat that movie as a 0 (average rating). Essentially this implies that the particular item is not contributing to the computation of r_{xi}.)

Once you have predicted the ratings that User ID 191 will give to the 4 movies, you will calculate the RMSE.

Your output should be of the following format:

```
User ID 191, 5 random user IDs: 375, 657, 513, 50, 225.
Using item-item similarity, User ID 191 will rate the movies as follows:
150: X.X
296: X.X
380: X.X
590: X.X
RMSE: Y.Y
```