CS 422: Data Mining

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

Fall 2021: Homework 9 (10 points)

Due date: Friday, December 03 2021 11:59:59 PM Chicago Time

NOTE CAREFULLY: This homework is due at the above date, LATE SUBMISSIONS WILL NOT BE HONORED AT ALL. Due to various constraints related to the finals week and grading the homework in time, late submissions will not be honored at all. Please submit your work on 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.

1. Exercises (4 points divided evenly among the questions) Please submit a PDF file containing answers to these questions. Any other file format that can be read will lead to a loss of 0.5 point. Non-PDF files that cannot be opened and read for grading will lead to a loss of all points allocated to this exercise.

1.1 Tan, Ch. 5 (Association Analysis)

Questions 1, 2, 9 (a), 9 (b), 15

1.2 Zaki, Chapter 8 (Frequent Pattern Mining)

Questions 1(a), 4

2. Practicum problems (6 points) 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 Association Analysis

In this assignment you will be using the *Extended Bakery* dataset, which describes transactions from a chain of bakery shops that sell a variety of drinks and baked goods.

The dataset is presented as a series of transactions, 1,000, 5,000, 20,000 and 75,000 transactions, stored in files named tr-1k.csv, tr-5k.csv, tr-20k.csv and tr-75k.csv, respectively. Each file contains the data in a sparse vector format, i.e., each line of the file has the following format:

1, 7, 15, 44, 49
2, 1, 19

...

The first column is the transaction ID and the subsequent columns contain a list of purchased goods from the bakery represented by their product ID code. In the example above, the first line implies that transaction ID one contained four items: 7, 15, 44, and 49. The mapping of the product ID to product name is provided in the products.csv file.

(a) **[1 points]** For each series of transaction files (i.e., tr-5k.csv, tr-20k.csv, …) create a canonical representation of the transaction file. A canonical representation for each dataset will be a file that contains a list of product names (not IDs) on a line, each product separated by a comma and a newline ends the line. So, as an example, the vector shown above would correspond to the following canonical representation:

Coffee Eclair, Blackberry Tart, Bottled Water, Single Espresso
Lemon Cake, Lemon Tart
…

Save the canonical representation in files with the canonical suffix, i.e., tr-5k-canonical.csv, and so on. Use these files for the rest of the work. **Include these files in the archive that you upload to Blackboard.**

You can use any programming language of your choice to do part (a).

(b) **[3 point]** Given the database of transactions, where each transaction is a list of items, find rules that associate the presence of one set of items with that of another set of items. Ideally, we only want to find rules that are substantiated by the data; we want to avoid spurious associations.

Find association rules that exceed specific values of *minimum support* and *minimum confidence*. You are free to experiment with different values until you find something that produces meaningful results. However, be aware that if you specify minimum support or confidence very low, your R process may appear to "hang" as the many itemsets are mined. In such a case, restart R. Use a structured approach by first reading the documentation to understand what the default value of *minsup* and *minconf* is, and then experiment from there.

Recall that finding rules requires two steps: finding the *frequent itemsets* and discovering strong *association rules* within them. You will use the R **arules** package as shown in class.

Your output should contain the following:

- For each frequent itemset:
    1. All items in it, described by the product names.
    2. The support of the itemset.
- For each rule:
    1. The antecedent.
    2. The consequent.
    3. The support of the rule.
    4. The confidence of the rule.

(c) **[1.5 points]** Given the above output, respond to the following question: Compare the rules you obtained for each different subset (1,000 – 75,000 transactions). How does the number of transactions affect the results you observed? (Write the answer in your R markup file, easily identified.)

(d) **[1.5 points]** Answer the following questions for the 75,000 transactions dataset using the same support level as determined in (b):

(i) What is the most frequently purchased item or itemset?

(ii) What is the least frequently purchased item or itemset?

2.2 Recommender Systems (Content-based recommendations) [6 points] EXTRA CREDIT

This is an EXTRA CREDIT problem. If you attempt the problem, and the answer is correct, you will awarded the 6 points as extra credit. In order to get the extra credit, the program must work completely, partial solutions will not be accepted.

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 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 on Blackboard (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	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	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