CS 498: Computational Advertising **Fall 2018**

Homework 3

Handed Out: October 17, 2018 *Due: October* 31, 2018 *11:59 pm*

1 General Instructions

- This assignment is due at 11:59 PM on the due date. We will be using Compass (http://compass2g.illinois.edu) for collecting the non-programming part of this assignment. Contact TAs if you face technical difficulties in submitting the assignment. We shall NOT accept any late submission!
- The non-programming part of homework MUST be submitted in pdf format. Handwritten answers are not acceptable. Name your pdf file as YourNetid-HW3.pdf
- For Questions 1 and 2, you need to explain the logic of your answer/result for every subpart. A result/answer without any explanation will not receive any points.
- The programming part of this assignment will be hosted on hackerrank (https://www.hackerrank.com/) as a programming contest. To participate in this contest, please open a hackerrank account with your illinois.edu email id. The contest framework will allow you to verify the correctness of your submission based on a set of sample test cases.
- It is OK to discuss with your classmates and your TAs regarding the methods, but it is NOT OK to work together or share code. Plagiarism is an academic violation to copy, to include text from other sources, including online sources, without proper citation. To get a better idea of what constitutes plagiarism, consult the CS Honor code (http://cs.illinois.edu/academics/honor-code) on academic integrity violations, including examples, and recommended penalties.There is a zero tolerance policy on academic integrity violations; Any student found to be violating this code will be subject to disciplinary action.
- Please use Piazza if you have questions about the homework. Also feel free to send TAs emails and come to office hours.
- Questions 1 and 2 are based on the following reference [1] chapter.

2 Question 1 (6 points)

Scroll to Section 9.8, Exercises in the reference chapter linked on the first page. In this question, you are required to solve Problem (3). In parts (a) and (b), you must show the precise steps to arrive at your answers. They carry 2 points each. Part (c) is also worth 2 points.

[1]https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book-ch09.pdf

3 Question 2 (4 points)

Scroll to Section 9.8, Exercises in the reference chapter linked on the first page. In this question, you are required to solve Problem (4). In parts (b), you must show the precise steps to arrive at your answers. Both parts carry 2 points each.

4 Programming Question (10 points)

The objective of this programming assignment is to design a movie recommender system. The main goal of such a system is to recommend relevant movies to an user based on available data. Data includes information about the movies and ratings provided by a user to a subset of movies. We will have some metadata information about each movie like title, a brief overview , tagline of the movie etc. We also have the ratings that a user has provided to some of the movies. Now based on movie metadata and ratings information, we need to recommend new movies to an user.

To recommend new movies to an user, we need to determine how relevant a movie is to an user. Relevance between a movie and an user is captured by the rating that a user provides to a particular movie. To recommend movies to an user, we need to infer/estimate the rating that a user would have provided to a movie if she had already seen this movie. So our recommendation problem becomes an *unknown user-movie rating estimation* problem.

Let the set of users be represented by $U(u \in U)$ and set of movies by represented by $M(m \in M)$. We want to estimate $\hat{r}_{um}$ i.e. rating provided by u to m. Note that for the (u, m) pairs already present in the data $\hat{r}_{um} = r_{um}$

For the (u, m) pairs not present in the data, we will use the formula below

$$\hat{r}_{um} = b_{um} + \frac{\sum_{j \in R(u)} s_{mj}(r_{uj} - b_{uj})}{\sum_{j \in R(u)} s_{mj}} \tag{1}$$

$$b_{um} = \mu + b_u + b_m \tag{2}$$

$$b_m = \frac{\sum_{u \in R(m)}(r_{um} - \mu)}{|R(m)|} \tag{3}$$

$$b_u = \frac{\sum_{m \in R(u)}(r_{um} - \mu - b_m)}{|R(u)|} \tag{4}$$

b_{um} is the baseline rating predictor. μ is the global mean calculated across all the ratings available in the dataset. $R(m)$ is the set of users that have rated the movie m in the available dataset. $R(u)$ are the set of the movies that have been rated by the user u. s_{mj} is the similarity value between 2 movies m and j.

We need the similarity value s_{mj} in Equation (1). In this assignment, we will use movie metadata content to calculate the value s_{mj} We will provide the movie metadata information as a document or collection of words. $s_{m,j}$ is the **cosine similarity** between the metadata document of the 2 movies. In order to calculate cosine similarity, first we need to convert the metadata documents to a vector form. The common way of doing this is to transform documents into **tf-idf** vector.

Term Frequency also known as **tf** measures the number of times a term (word) occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is normalized by the document length.

$$tf(t, m) = \frac{\text{Number of times term t appears in metadata of a movie m}}{\text{Total number of terms in the metadata about m}}$$

Inverse Document Frequency also known as **idf** measures how important a term is to a particular movie.

$$idf(t) = \log_e(\frac{\text{Total number of movies in dataset}}{\text{Number of movies with term t in it}})$$

Now let the total number of unique words across all movie metadata documents be $V = (V_1, V_2, ..., V_{|V|})$. Now we will convert metadata document for movie m into a $|V|$ dimensional vector d^m.

$$d^m = (d_1^m, d_2^m, ..., d_{|V|}^m)$$
$$d_i^m = tf(V_i, m) * idf(V_i)$$

Now we will calculate the similarity as

$$s_{mj} = cosine(d^m, d^j)$$

In this MP given an input of a (u, m) , you need to output the estimated rating value $\hat{r}_{um}$. Round $\hat{r}_{um}$ to 1 decimal point.