Big Data Analytics

ESSEC

Home work 7: Recommendation Systems

1. **Exercise 9.2.1 MMDS book:** Three computers, A, B, and C, have the numerical features listed below:

Feature	A	B	C
Processor Speed	3.06	2.68	2.92
Disk Size	500	320	640
Main-Memory Size	6	4	6

We may imagine these values as defining a vector for each computer; for instance, A's vector is $[3.06, 500, 6]$. We can compute the cosine distance between any two of the vectors, but if we do not scale the components, then the disk size will dominate and make differences in the other components essentially invisible. Let us use 1 as the scale factor for processor speed, α for the disk size, and β for the main memory size.

   (a) In terms of α and β, compute the cosines of the angles between the vectors for each pair of the three computers.

   (b) What are the angles between the vectors if $\alpha = \beta = 1$?

   (c) What are the angles between the vectors if $\alpha = 0.01$ and $\beta = 0.5$?

   (d) A certain user has rated the three computers as follows: A: 4 stars, B: 2 stars, C: 5 stars.

   - Normalize the ratings for this user.
   - Compute a user profile for the user, with components for processor speed, disk size, and main memory size.

2. **Exercise 9.3.1 MMDS book:** The following Figure is a utility matrix, representing the ratings, on a 1–5 star scale, of eight items, a through h, by three users A, B, and C.

	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

Compute the following from the data of this matrix.

   (a) Treating the utility matrix as boolean, compute the Jaccard distance between each pair of users.

   (b) Repeat Part (a), but use the cosine distance.

   (c) Treat ratings of 3, 4, and 5 as 1 and 1, 2, and blank as 0. Compute the Jaccard distance between each pair of users.

   (d) Repeat Part (c), but use the cosine distance.

(e) Normalize the matrix by subtracting from each nonblank entry the average value for its user.

(f) Using the normalized matrix from Part (e), compute the cosine distance between each pair of users.

Spark

Write a Spark program that implements a simple "People You Might Know" social network friendship recommendation algorithm. The key idea is that if two people have a lot of mutual friends, then the system should recommend that they connect with each other. Download the input file *'soc-LiveJournal1Adj.tx'* from the Moodle. The input file contains the adjacency list and has multiple lines in the following format: $< User >< TAB >< Friends >$. Here, $< User >$ is a unique integer ID corresponding to a unique user and $< Friends >$ is a comma separated list of unique IDs corresponding to the friends of the user with the unique ID $< User >$. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A. The data provided is consistent with that rule as there is an explicit entry for each side of each edge.

Algorithm: Let us use a simple algorithm such that, for each user U, the algorithm recommends $N = 10$ users who are not already friends with U, but have the most number of mutual friends in common with U.

Output The output should contain one line per user in the following format: $< User >< TAB >< Recommendations >$ where $< User >$ is a unique ID corresponding to a user and $< Recommendations >$ is a comma separated list of unique IDs corresponding to the algorithm's recommendation of people that $< User >$ might know, ordered in decreasing number of mutual friends. Even if a user has less than 10 second-degree friends, output all of them in decreasing order of the number of mutual friends. If a user has no friends, you can provide an empty list of recommendations. If there are recommended users with the same number of mutual friends, then output those user IDs in numerically ascending order.

Tips: The default memory assigned to the Spark runtime may not be enough to process this data file, depending on how you write your algorithm. Then you'll need to increase the memory assigned to the Spark runtime to 8GB.