Assignment 2 Part 1: Algorithm Description

Friend Recommendation using Mutual Friends

Algorithm description:

1. Identify Potential Friends: For each user in the dataset, find potential friends by exploring the friends of the user's friends. Exclude the user's current friends and the user themselves from this list.
2. Filter Target Users: Focus on the list of 10 target users for whom we need to generate friend suggestions.
3. Count Mutual Friends: For each target user and their potential friends, count the number of mutual friends they share.
4. Aggregate Potential Friends: Combine the counts of mutual friends for each target user and their potential friends into a single list for each user.
5. Sort and Select Top 10: Sort the list of potential friends for each target user based on the number of mutual friends in descending order. Select the top 10 potential friends.
6. Display Results: Present the results in a structured format.

Detailed Algorithm description:

Input:

- df: DataFrame in the format ["userid": [list_of_friends]]
- user_list: List of 10 user IDs for which friend suggestions are to be generated.
- data_dict: Broadcast variable containing the mapping of userid to their respective friends.

Steps:

1. *FlatMap Operation:*
   For each user x and their list of friends x[1], generate pairs (user, friend) where:
   - user is the original user x[0].
   - friend is a potential friend from the friends of friends who is not already a friend and not the user themselves.

   The output is a tuple ((user, potential_friend), 1).
2. *Filter Operation:*
   Keep only those pairs where the user is in the user_list (the list of 10 users for whom recommendations are to be generated).
3. *ReduceByKey Operation:*
   Sum the mutual friend counts for each ((user, potential_friend), count) pair, resulting in the total number of mutual friends for each potential friendship.
4. *Map Operation:*
   Transform the reduced RDD to ((user), [(potential_friend, count)]), where the value is a list of tuples containing potential friends and their respective mutual friend counts.
5. *ReduceByKey Operation:*
   Aggregate the lists of tuples for each user, resulting in one list of potential friends with mutual friend counts per user.
6. *SortByKey Operation:*

Sort the RDD by the user key to ensure the results are ordered by user ID.

7. *Map Operation:*
   Convert the RDD to a dictionary format with keys “user” and “recommendations”:
   - “user” contains the user ID.
   - “recommendations” contains a sorted list of potential friends based on the count of mutual friends (in descending order), limited to the top 10 suggestions.
8. *Map Operation:*
   Transform the dictionary to a tuple format:
   - User ID.
   - A comma-separated string of the top 10 recommended friend IDs.
   - A comma-separated string of the top 10 recommended friends along with their mutual friend counts in the format friend: count.
9. *SortBy Operation:*
   Sort the final RDD by the user ID to ensure the results are in order.
10. *Convert to DataFrame and Display:*
    Convert the RDD to a DataFrame with columns “User”, “Recommendations”, and “Mutuals Count”.

Output:

Table

	User	Recommendations	Mutuals Count
1	0	38737, 18591, 27383, 34211, 1532, 12143, 12561, 17880, 22939, 25212	38737: 5, 18591: 4, 27383: 4, 34211: 4, 12143: 3, 12561
2	31	29696, 29704, 29695, 29703, 29709, 29692, 29712, 41079, 27383, 29710	29696: 7, 29704: 7, 29695: 5, 29703: 5, 29709: 5, 29692
3	233	95, 213, 240, 6935, 164, 17150, 19388, 20590, 33621, 38741	95: 7, 213: 5, 240: 4, 6935: 3, 164: 2, 17150: 2, 19388: 2
4	1234	1230, 3895, 15186, 19211, 19427, 23202, 23690, 27549, 27609, 27736	1230: 3, 15186: 3, 19211: 3, 19427: 3, 23202: 3, 23690:
5	7584	7530, 7580, 7529, 7537, 7564, 7581, 7596, 7603, 5667, 5684	7530: 9, 7580: 7, 7529: 5, 7537: 5, 7564: 5, 7581: 5, 759
6	8675	239, 16883, 16862, 15356, 16914, 16966, 17022, 17740, 16896, 16898	239: 17, 16883: 15, 16862: 11, 15356: 8, 16914: 8, 1696
7	18853	15715, 18852, 11066, 11863, 11865, 12447, 12715, 13637, 13640, 13648	15715: 2, 18852: 2, 11066: 1, 11863: 1, 11865: 1, 12447
8	21212	21184, 1230, 1357, 11005, 11190, 11707, 13106, 13223, 13279, 13423	21184: 2, 11005: 1, 11190: 1, 11707: 1, 1230: 1, 13106:
9	36654	36657, 36658, 12339, 12991, 22049, 24336, 28381, 30842, 30870, 30920	36657: 2, 36658: 2, 12339: 1, 12991: 1, 22049: 1, 24336
10	49998	11377, 34439, 34450, 45133, 11383, 11385, 13852, 13863, 23510, 27555	11377: 4, 34439: 3, 34450: 3, 45133: 3, 11383: 2, 11385

10 rows | 14.69 seconds runtime

Refreshed now