CSC 582/782: Big Data

Assignment 3

Finding Common Friends using MapReduce

(50 points)

Due: Sunday, November 20

Problem Statements

Given that a social network with a large number of users such as Facebook, Facebook contains a list of friends and the friends relationship are bi-directional (If *Alice* is *Tom's* friend, *Tom* is also *Alice's* friend). When you visit someone's profile in Facebook, you can look at a list of friends that you have in common. For example, this feature shows that "You and Mike have 50 friends in common". This is one of the common requests serviced by the Facebook. Actually, hundreds of millions of these requests are made by users every day so that the Facebook decided to reduce the processing time of the request. Because it is not often that the list of friends changes, it is waste of time to recalculate whenever the user visits someone's profile.

In order to improve time computation of the request, there are many sophisticated strategies (e.g., a caching, MapReduce, etc.) to find common friends between users of the social network site. The social network sites use a MapReduce solution which calculates all the user's common friends once for a day and stores Hadoop results to the disk. Since the disk cost is relatively cheap, the lookup operation to the results on the disk is efficient when the request is made.

So, **Write a MapReduce program to find common friends (a.k.a. mutual friends) amongst all pairs of users.** Suppose that we have a set of social network users: $\{User_1, User_2, \dots, User_n\}$. The goal of the project is to find common friends for every pair of $(User_i, User_j)$ and $i \neq j$.

Program Requirements

- Input

The input file should contain the following format:

<UserID$_i$><,><FriendID$_1$><space><FriendID$_2$>…< FriendID$_m$>

where UserID$_i$ represents a unique user ID and $\sum_{j=1}^{m}(\text{FriendID}_j)$ are friends of the UserID$_i$. Note that a FriendID is one of UserIDs. For example,

101,102 103 104 105 106

102,101 103 104
103,101 102 104 105
104,101 102 103
105,101 103
106,101

In the example above, the UserID 106 has only one friend (i.e., UserID 101) but UserID 105 has two friends: UserID 101 and 103.

- Output
After running your MapReduce program in Hadoop, the output will follow the format below:

<UserID_i><,><UserID_j><space><[><FriendID_1><,><FriendID_2>….< FriendID_m><]>

From the same example above,
101,102 [103,104]
101,103 [102,104,105]
101,104 [102,103]
101,105 [103]
101,106 []
102,103 [101,104]
102,104 [101,103]
103,104 [101,102]
103,105 [101]

In the generated output, we can see that the UserIDs 101 and 103 have common friends of UsersID 102, 104 and 105. However, UserIDs 101 and 106 don't have common friends. Also, there is a simple input file you can use: **sample.txt (small dataset)** and **sample2.txt (large dataset)**.

- Pseudo codes (Map/Reduce function)

In Mapper,
Map() function
//(input_key, input_value) = ($UserID_i$, a list of associated FriendIDs for the $UserID_i$)
//input_key = $UserID_i$
//input_value = ($FriendID_1$ $FriendID_2$ … $FriendID_m$)

```
map( input_key, input_value) {
    reducer_value = (FriendID_1 FriendID_2 ... FriendID_m)
    for each FriendID in (FriendID_1 FriendID_2 ... FriendID_m) {
        reducer_key = build_sorted_key(UserID_i, FriendID) //prevent duplicate keys to be generated
        //reducer_key = a tuple(UserID_i, FriendID_j) //create a tuple where FriendID_j ∈ UserID_i
        //reducer_value = a list of all friends for the UserID_i (it's same as input_value)
        emit( reducer_key, reducer_value)
```

```
    }
}
```

In Reducer,

//Prior to sending the key-value pairs to the reducers, they should be grouped by keys.

Reduce() function
//reducer_key = a tuple($UserID_i$*,* $UserID_k$*) where* $UserID_k \in FriendID$
//reducer_value = List {$List_1$*,* $List_2$*, ...,* $List_n$*},*
//where each $List_i$ *= {set of unique friendIDs corresponding the reducer_key }*

```
reduce(reducer_key, reducer_value) {
      output_key = reducer_key;
      //get an intersection between all sets of friends to find common friends
      // for the pair of (UserID_i, UserID_k)
      output_value = intersection (List_1, List_2, ..., List_n);
      //output_key = a pair of (UserID_i, UserID_k)
      //output_value = [a list of common friends]
      emit(output_key, output_value);
}
```

Note: Although the pseudo code is shown in the program requirement, this might be a starting point to help out write your program. If you have a better logic than the given pseudo code or optimize it, I am always welcome. But, you should mention your own logic in Report.

Programming Requirements

You are required to use Python for this assignment. Also, your program should compile and run on your virtual hadoop cluster from the first assignment.

Submission Requirements

The assignment must be done **individually**. Please submit a single zip (or tar) file that include all files. The submission should include:

- **Source codes**: your Python codes for MapReduce as well as shell script files (if applicable) that you use for testing. All programs (source code) must contain comments. Failure in using proper comments will result in reduced points assigned to the programming assignments.
- **Report:** A short summary on instruction to run your program and the provide a screenshot as testing results by your program. Note that screenshots must be readable for full credit.
- **Dataset**: a dataset for your results

Late Policy

Please check the late policy on the course syllabus.