Ch. Anusha
22PA1A4223
AIML-A

ASSIGNMENT-1

Implement Friends-of-friends algorithm in Map reduce

Hint: Two Map reduce jobs are required to calculate the FOFs for each user in a social network - The first job calculates the common friends for each user, & second job sorts the common friends by the number of connections to your friends.

The friends of friends algorithm is used to suggest new friends in a social network based on mutual connection - The goal is to

(1) Identify mutual friends between users
(2) Count the number of mutual connections
(3) Sort and rank friend suggestions based on the number of mutual friends.

To achieve this efficiently for large datasets, we use Map reduce, which consists of two Map reduce Jobs.

Two-Stage Map reduce Approach

Map reduce Job1: Generating Mutual friends.

- Mapper:
  - Reades friendship data
  - emits direct friendship data
  - Emits mutual friend pais for potential recommendation.
- Reducer:
  - Filters out direct friends
  - Emits mutual friends suggestions

Map reduce Job2: Ranking Friends

- Mapper:
  Groups mutual friends counts
- Reducer:
  Sorts friend Suggestions by the number of mutual friends

program):

This Script is designed to run in using Hadoop's Mapreduce framework with mrjob

Step 1: Implement Mapreduce Job

```
from mrjob.job import MRJob
from mrjob.step import MRStep
from itertools import combinations.
Class friendoffriend (MRJOB):
    def steps (self):
        return[
            MRStep (mapper = self.mapper_ extract_friends,
                reducer = self.reducer_ generate_mutual_friends),
            MRStep (mapper = self.mapper_count_ mutualfriends,
                reducer = Self.reducer_sort_ Suggestions)
        ]

    def mapper_extract=friends(self,_, line)
        user, friends = line.split (':')
        user = user.strip()
        friends= friends. strip(). Split (',')
```

```
for friend in friends:
    yield (user, friend), 'direct'

for friend1, friend2 in combinations (friends, 2):
    yield (friend1, friend2), 'mutual'
    yield (friend2, friend1), 'mutual'

def reducer_generate_mutual_friends (self, key, values):
    values_list = list(values)
    if 'direct' in values_list:
        return
    yield key, 1

def mapper_count_mutual_friends (self, key, valuecount)
    user1, user2 = key
    yield user1, (user2, count)
    yield user2, (user1, count)

def reducer_sort_suggestions (self, user, friend_counts):
    sorted_friends = sorted(friend_counts, key = lambda
                                                x: -x[1])
    yield user, sorted_friends

if __name__ == "__main__"
    FriendsOfFriends.run()
```

Step 2 : Input data format

A : B, C, D

B : A, C, E

C : A, B, D, E

D : A, C, E

E : B, C, D

Step 3: Running the Mapreduce job

To run the script on Local input

```
python fof-mapreduce.py friends.txt
```

If running on Hadoop, use:

```
hadoop jar /path/to/hadoop-streaming.jar -input friends.txt
    -output fof-output- mapper mapper.py,
    -reducer reducer.py
```

Step 4: Expected output

"A" [("C", 2)]

"B" [("D", 2)]

"C" [("B", 2)]

"D" [("B", 2)]

"E" [("A", 2)]

Complexity Analysis:

1. Job1 Mapper (Extract Friendships) — O(NXF)
2. Job1 Reducer (Count Mutuals) — O(NXF²) Worst case
3. Job2 Mapper (Group by users) — O(NXM)
4. Job2 reducer (Sort recommendations) — O(N log M)