EECS E6893 Big Data Analytics
HW2: Friend Recommendations, GraphFrames

Hritik Jain, hj2533@columbia.edu

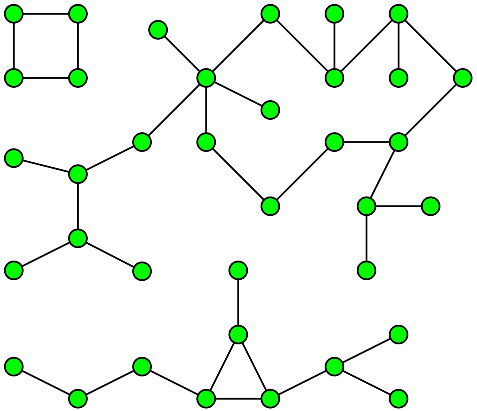
11/13/2020

GraphFrames

- DataFrame-based Graph
- GraphX is to RDDs as GraphFrames are to DataFrames
- Represent graphs: vertices (e.g. users) and edges (e.g. relationships between users)
- GraphFrames package separate from core Apache Spark

Connected components

- A subgraph where any two vertices are connected to each other by edges, but not connected to other vertices in the graph
- In a social network, connected components can approximate clusters
- In the GraphFrame, the connected components algorithm labels each connected component of the graph with the ID of its lowest-numbered vertex



Reference: https://en.wikipedia.org/wiki/Component_(graph_theory)

PageRank

- PageRank measures the importance of each vertex in a graph
- An edge from *u* to *v* represents an endorsement of *v*'s importance by *u*

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

d: damping factor;

default = 0.85 - 15% chance that a typical users won't follow any links on the page and instead navigate to a new random URL.

- Convergence occurs when all PageRank values are within the margin of error.

Reference: https://en.wikipedia.org/wiki/PageRank

PageRank (Spark)

```
pageRank(resetProbability=0.15, sourceId=None, maxIter=None, tol=None)
```

Parameters:

resetProbability: 1-d, Probability of resetting to a random vertex, default=0.15

maxIter: If set, the algorithm is run for a fixed number of iterations.

tol: If set, the algorithm is run until the given tolerance/margin of error.

NOTE: Exactly one of *maxIter* or *tol* must be set.

HW2

- Question 1: Friend Recommendations
- Question 2: Graph Analysis using GraphFrames

Environment Setup

1. Create multiple workers on Dataproc instead of single node, otherwise it will take long time to run.
2. Install graphframe package in spark when create the cluster.

   (You can reference to config Spark properties)

Cloud Shell:

This is for Python 3.
You can modify it.

```
   gcloud beta dataproc clusters create <cluster-name>
   --optional-components=ANACONDA,JUPYTER --image-version=preview
   --enable-component-gateway --bucket <bucket-name> --project <project-id>
1. --num-workers 3  --metadata PIP_PACKAGES=graphframes==0.6
   --initialization-actions
   gs://dataproc-initialization-actions/python/pip-install.sh
2. --properties
   spark:spark.jars.packages=graphframes:graphframes:0.6.0-spark2.3-s 2.11
```

Q1

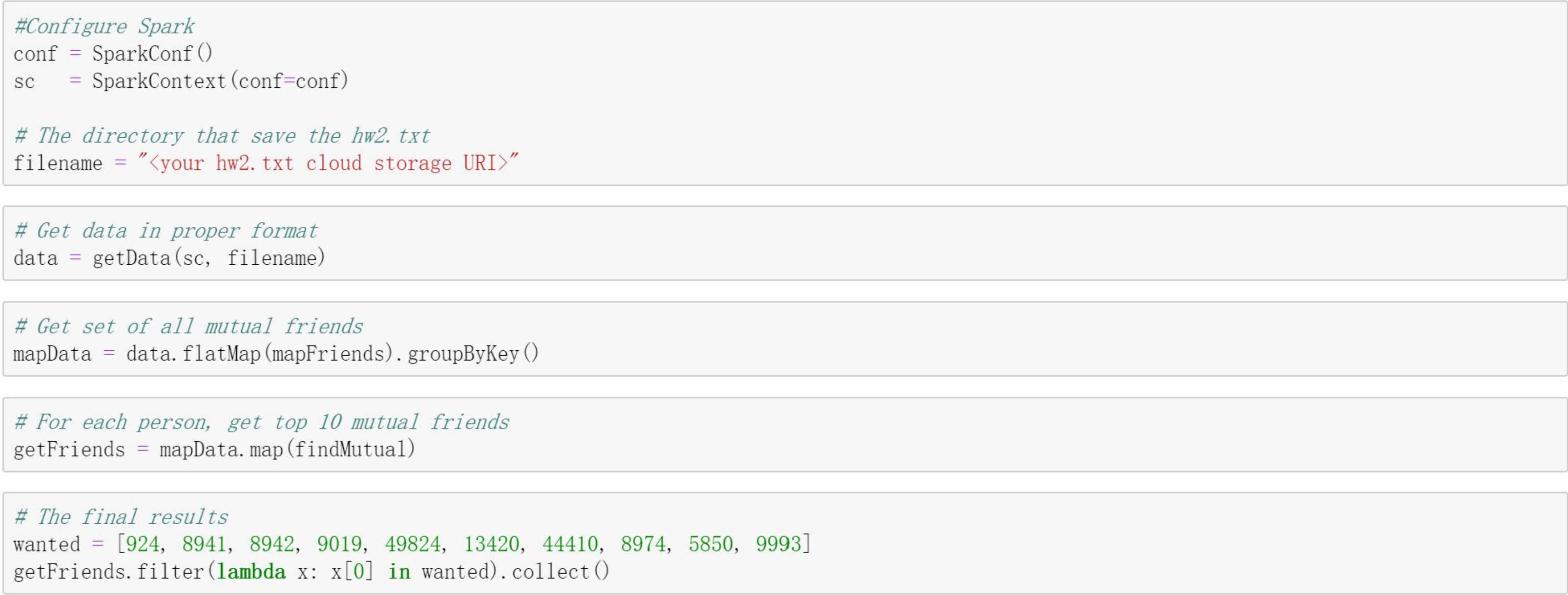
- Write a Spark program that implements a simple “People You Might Know” social network friendship recommendation algorithm. The key idea is that if two people have a lot of mutual friends, then the system should recommend that they connect with each other.
- Question: Give recommendation for 10 Users

- Dataset Format

  <User> <Tab> <Friends>

  <User> is a unique ID ; <Friends> are comma separated list of unique IDs

Q1 - Code Skeleton



```
#Configure Spark
conf = SparkConf()
sc   = SparkContext(conf=conf)

# The directory that save the hw2.txt
filename = "<your hw2.txt cloud storage URI>"
```

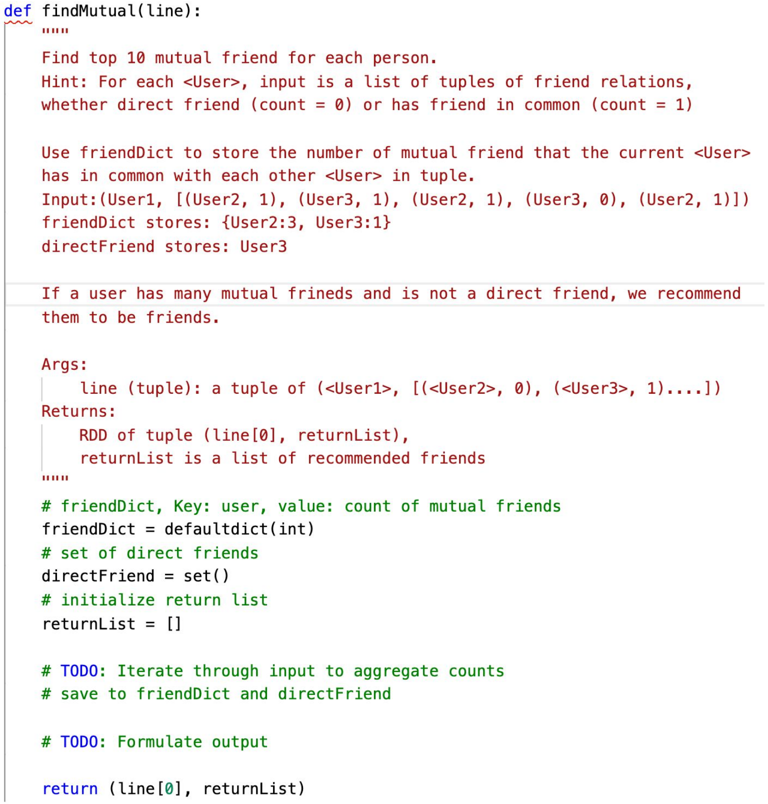
```
# Get data in proper format
data = getData(sc, filename)
```

```
# Get set of all mutual friends
mapData = data.flatMap(mapFriends).groupByKey()
```

```
# For each person, get top 10 mutual friends
getFriends = mapData.map(findMutual)
```

```
# The final results
wanted = [924, 8941, 8942, 9019, 49824, 13420, 44410, 8974, 5850, 9993]
getFriends.filter(lambda x: x[0] in wanted).collect()
```

Q1 - Function example



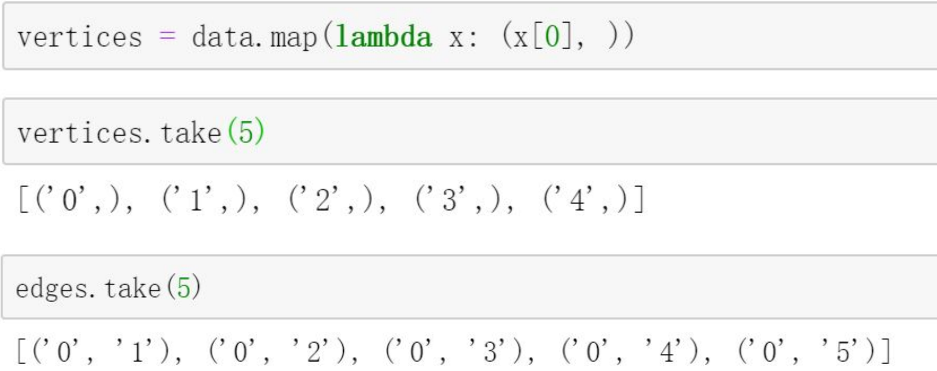
```
def findMutual(line):
    """
    Find top 10 mutual friend for each person.
    Hint: For each <User>, input is a list of tuples of friend relations,
    whether direct friend (count = 0) or has friend in common (count = 1)

    Use friendDict to store the number of mutual friend that the current <User>
    has in common with each other <User> in tuple.
    Input:(User1, [(User2, 1), (User3, 1), (User2, 1), (User3, 0), (User2, 1)])
    friendDict stores: {User2:3, User3:1}
    directFriend stores: User3

    If a user has many mutual frineds and is not a direct friend, we recommend
    them to be friends.

    Args:
        line (tuple): a tuple of (<User1>, [(<User2>, 0), (<User3>, 1)....])
    Returns:
        RDD of tuple (line[0], returnList),
        returnList is a list of recommended friends
    """
    # friendDict, Key: user, value: count of mutual friends
    friendDict = defaultdict(int)
    # set of direct friends
    directFriend = set()
    # initialize return list
    returnList = []

    # TODO: Iterate through input to aggregate counts
    # save to friendDict and directFriend

    # TODO: Formulate output

    return (line[0], returnList)
```

Q2

- Use the Q1 dataset again do the graph analysis
- Connected Component
- PageRank

Q2

- Steps 1
  - Format the provided dataset into two Spark DataFrames: edges and vertices
    - Notice: For the vertices, if there is no other properties for vertices (like in our case), then we should create tuple like this, otherwise a string inside a tuple will not be identified as a tuple but as a single string. If there are other properties, then no need for that extra comma.



```
vertices = data.map(lambda x: (x[0], ))
```

```
vertices.take(5)
```

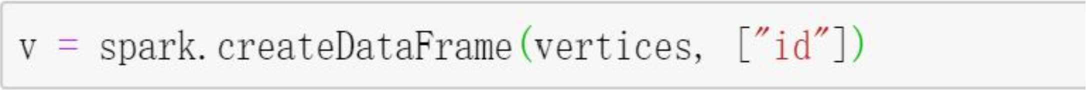
```
[(’0’,), (’1’,), (’2’,), (’3’,), (’4’,)]
```

```
edges.take(5)
```

```
[(’0’, ’1’), (’0’, ’2’), (’0’, ’3’), (’0’, ’4’), (’0’, ’5’)]
```

Q2

- Step 2
  - Convert the RDD to DataFrame
    - Directly convert to DataFrame
    - Save RDD to csv, then read csv to DataFrame



```
v = spark.createDataFrame(vertices, [”id”])
```

- Step 3
  - Create graph

```
from graphframes import *

g = GraphFrame(v, e)
```

If you set the environment correctly following the instructions above, there should be no problem with Jupyter.

If you are using Spark shell and it doesn’t work, you could try running:

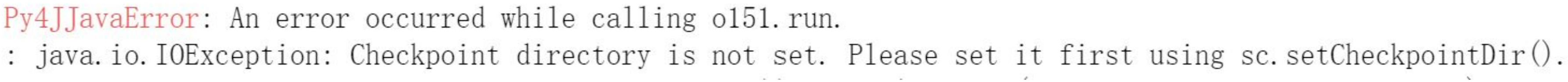
```
pyspark --packages
graphframes:graphframes:0.6.0-spark2.3-s_2.11
```

running Spark

Q2 - Connected components

- Notice

  If you are using Connected components, and get the error like



```
Py4JJavaError: An error occurred while calling o151.run.
: java.io.IOException: Checkpoint directory is not set. Please set it first using sc.setCheckpointDir().
```

  You could reference the following answer on stackoverflow

  https://stackoverflow.com/questions/49159896/how-to-set-checkpiont-dir-pyspark-data-science-experience

Q2 - PageRank

- results = g.pageRank(resetProbability=0.15, tol=0.01)
- There are multiple parameters. You can play with them, see whether there are different result.

References

- https://graphframes.github.io/graphframes/docs/_site/index.html
- https://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm