Graphs

May 22, 2018

1 CIS 545 Homework 2: Graphs

```
In [1]: # Execute this once, the first time you run
        !pip install networkx

        # Disable Python warning messages - you should probably only run this before submission

        import warnings
        warnings.filterwarnings('ignore')

Requirement already satisfied: networkx in /opt/conda/lib/python3.6/site-packages
Requirement already satisfied: decorator>=4.1.0 in /opt/conda/lib/python3.6/site-packages (from
```

1.1 Step 2.1 Spark Setup

```
In [2]: # TODO: Connect to Spark as per Step 2.1
        from pyspark.sql import SparkSession
        from pyspark.sql.types import *
        import pyspark.sql.functions as F

        spark = SparkSession.builder.appName('Graphs-HW2').getOrCreate()

In [3]: # Load some dummy data, which should be overwritten in Step 2.2

        answers_sdf = spark.createDataFrame([{'from_node': 123, 'to_node': 456},\
                                            {'from_node': 456, 'to_node': 789},
                                            {'from_node': 456, 'to_node': 890}])
        comments_answers_sdf = spark.createDataFrame([{'from_node': 123, 'to_node': 456}])
        comments_questions_sdf = spark.createDataFrame([{'from_node': 123, 'to_node': 456}])

        graph_sdf = spark.createDataFrame([{'from_node': 123, 'to_node': 456}])
```

1.2 Step 2.2 Loading

```
In [4]: # TODO: load data as per Step 2.2
        answers_sdf = spark.read.load('sx-stackoverflow-a2q.txt', format="text")
        comments_answers_sdf = spark.read.load('sx-stackoverflow-c2a.txt', format="text")
        comments_questions_sdf = spark.read.load('sx-stackoverflow-c2q.txt', format="text")
```

```
In [5]: # You may add as many cells as you like here.
        # Use Insert | Insert Cell Below
```

1.3 Step 2.2 Results

```
In [6]: answers_sdf.count()
```

```
Out[6]: 17823525
```

```
In [7]: answers_sdf.show()
```

```
+----------------+
|           value|
+----------------+
|  9 8 1217567877|
|  1 1 1217573801|
| 13 1 1217606247|
| 17 1 1217617639|
| 48 2 1217618182|
| 17 1 1217618239|
| 19 9 1217618357|
|13 23 1217618560|
|13 11 1217618799|
|23 23 1217619360|
|35 33 1217620542|
|39 33 1217620597|
|43 22 1217620971|
|17 32 1217621272|
|39 40 1217621416|
|37 40 1217621670|
|45 45 1217621917|
|17 17 1217622124|
|49 13 1217623079|
|13 23 1217623216|
+----------------+
only showing top 20 rows
```

```
In [8]: answers_sdf.printSchema()
```

```
root
 |-- value: string (nullable = true)
```

```
In [9]: comments_answers_sdf.count()
```

```
Out[9]: 25405374

In [10]: comments_answers_sdf.show(10)

+--------------------+
|               value|
+--------------------+
|     1 91 1220713630|
|     3 91 1220713792|
|   380 350 1220715736|
|4642 2257 1220734307|
|4642 1324220 1220...|
|2495 4285 1220736640|
|4642 4893 1220737355|
|2515 4903 1220738560|
|2515 4893 1220739071|
|   199 199 1220741079|
+--------------------+
only showing top 10 rows



In [11]: comments_answers_sdf.printSchema()

root
 |-- value: string (nullable = true)



In [12]: comments_questions_sdf.count()

Out[12]: 20268151

In [13]: comments_questions_sdf.show(10)

+--------------------+
|               value|
+--------------------+
|4550 4550 1220729190|
|   242 184 1220733503|
|4213 4946 1220768149|
|    91 91 1220768295|
|2658 1874 1220771891|
|4035 1874 1220773037|
|2257 4489 1220802041|
|   577 577 1220834891|
|4489 4489 1220853536|
|  828 2783 1220854143|
+--------------------+
only showing top 10 rows
```

```
In [14]: comments_questions_sdf.printSchema()
```

```
root
 |-- value: string (nullable = true)
```

1.4 Step 2.3

```
In [15]: # TODO: wrangling work in Step 2.3.  Add as many Cells as you need
         #answers_sdf.createOrReplaceTempView('answers_view')
         #answers_sdf=spark.sql('select split(value," ")[0] As from_node, split(value," ")[1] As
         answers_sdf=answers_sdf.withColumn('new_col',F.lit('answer'))
         answers_sdf1=answers_sdf.select(F.split(answers_sdf.value,' ')[0].alias('from_node').ca
         answers_sdf1=answers_sdf1.select(answers_sdf1.from_node,answers_sdf1.to_node,answers_sd
         #answers_sdf1.show(6)
         answers_sdf=answers_sdf1
         ## Step 2.3 Results
```

```
In [16]: #answers_sdf.createOrReplaceTempView('answers_view')
         comments_answers_sdf=comments_answers_sdf.withColumn('new_col',F.lit('comment-on-answer
         comments_answers_sdf1=comments_answers_sdf.select(F.split(comments_answers_sdf.value,'
         comments_answers_sdf1=comments_answers_sdf1.select(comments_answers_sdf1.from_node,comm
         #comments_answers_sdf1.show(6)
         comments_answers_sdf=comments_answers_sdf1
```

```
In [17]: #answers_sdf.createOrReplaceTempView('answers_view')
         comments_questions_sdf=comments_questions_sdf.withColumn('new_col',F.lit('comment-on-qu
         comments_questions_sdf1=comments_questions_sdf.select(F.split(comments_questions_sdf.va
         comments_questions_sdf1=comments_questions_sdf1.select(comments_questions_sdf1.from_nod
         #comments_questions_sdf1.show(6)
         comments_questions_sdf=comments_questions_sdf1
```

```
In [18]: graph_sdf=answers_sdf.unionAll(comments_questions_sdf).unionAll(comments_answers_sdf)
```

```
In [19]: answers_sdf.count()
```

```
Out[19]: 17823525
```

```
In [20]: answers_sdf.show(5)
```

```
+---------+-------+---------+
|from_node|to_node|edge_type|
+---------+-------+---------+
|        9|      8|   answer|
|        1|      1|   answer|
|       13|      1|   answer|
|       17|      1|   answer|
|       48|      2|   answer|
+---------+-------+---------+
only showing top 5 rows
```

```
In [21]: answers_sdf.printSchema()
```

```
root
 |-- from_node: integer (nullable = true)
 |-- to_node: integer (nullable = true)
 |-- edge_type: string (nullable = false)
```

```
In [22]: comments_answers_sdf.count()
```

```
Out[22]: 25405374
```

```
In [23]: comments_answers_sdf.show(5)
```

```
+---------+-------+-----------------+
|from_node|to_node|        edge_type|
+---------+-------+-----------------+
|        1|     91|comment-on-answer|
|        3|     91|comment-on-answer|
|      380|    350|comment-on-answer|
|     4642|   2257|comment-on-answer|
|     4642|1324220|comment-on-answer|
+---------+-------+-----------------+
only showing top 5 rows
```

```
In [24]: comments_answers_sdf.printSchema()
```

```
root
 |-- from_node: integer (nullable = true)
 |-- to_node: integer (nullable = true)
 |-- edge_type: string (nullable = false)
```

```
In [25]:  comments_questions_sdf.count()
```

```
Out[25]: 20268151
```

```
In [26]: comments_questions_sdf.show(5)
```

```
+---------+-------+-------------------+
|from_node|to_node|          edge_type|
+---------+-------+-------------------+
|     4550|   4550|comment-on-question|
|      242|    184|comment-on-question|
|     4213|   4946|comment-on-question|
|       91|     91|comment-on-question|
|     2658|   1874|comment-on-question|
+---------+-------+-------------------+
only showing top 5 rows
```

```
In [27]: comments_questions_sdf.printSchema()
```

```
root
 |-- from_node: integer (nullable = true)
 |-- to_node: integer (nullable = true)
 |-- edge_type: string (nullable = false)
```

```
In [28]: graph_sdf.count()
```

```
Out[28]: 63497050
```

```
In [29]: graph_sdf.show(5)
```

```
+---------+-------+---------+
|from_node|to_node|edge_type|
+---------+-------+---------+
|        9|      8|   answer|
|        1|      1|   answer|
|       13|      1|   answer|
|       17|      1|   answer|
|       48|      2|   answer|
+---------+-------+---------+
only showing top 5 rows
```

```
In [30]: graph_sdf.printSchema()
```

```
root
 |-- from_node: integer (nullable = true)
 |-- to_node: integer (nullable = true)
 |-- edge_type: string (nullable = false)
```

1.5 Step 2.4

```
In [31]: # You may put any computations you need here
         no_of_questions_sdf=answers_sdf.unionAll(comments_questions_sdf)
         no_of_questions_sdf=no_of_questions_sdf.groupBy(no_of_questions_sdf['to_node'].alias('u
         no_of_questions_sdf=no_of_questions_sdf.orderBy('count',ascending=False)
         #no_of_questions_sdf.select(no_of_questions_sdf['to_node'].alias('user')).show(5)
```

```
In [32]: no_of_answers_sdf=answers_sdf[(answers_sdf.from_node)!=(answers_sdf.to_node)]
         no_of_answers_sdf=no_of_answers_sdf.groupBy(no_of_answers_sdf['from_node'].alias('user'
         no_of_answers_sdf=no_of_answers_sdf.orderBy('count',ascending=False)
```

```
In [33]: list_op=(comments_questions_sdf.select('to_node').subtract(answers_sdf.select('to_node'
```

```
In [34]: #sdf1=answers_sdf.groupBy(answers_sdf.from_node,answers_sdf.to_node).count()
         #sdf2=sdf1
         #sdf1.join(sdf2,(F.col(sdf1.to_node)==(F.col(sdf2.from_node))),(F.col(sdf1.to_node)==(F
         answers_sdf.groupBy("from_node", "to_node").count()


         #(answers_sdf.groupBy("from_node", "to_node").count().as("count1")).
              # join(answers_sdf.groupBy("col1").agg(sum($"col3").as("sum_level1")), Seq("col1")

Out[34]: DataFrame[from_node: int, to_node: int, count: bigint]
```

```
In [35]: answers_sdf.show(10)

+---------+-------+---------+
|from_node|to_node|edge_type|
+---------+-------+---------+
|        9|      8|   answer|
|        1|      1|   answer|
|       13|      1|   answer|
|       17|      1|   answer|
|       48|      2|   answer|
|       17|      1|   answer|
|       19|      9|   answer|
|       13|     23|   answer|
|       13|     11|   answer|
|       23|     23|   answer|
+---------+-------+---------+
only showing top 10 rows
```

1.6 Step 2.4.1 Results

```
In [36]: # TODO: output dataframe with top 10 users by number of questions
         no_of_questions_sdf.show(10)
```

```
+------+-----+
|  user|count|
+------+-----+
|875317| 7995|
|  4653| 7652|
| 34537| 6991|
|117700| 6830|
| 39677| 6686|
|541686| 5618|
| 84201| 5528|
|859154| 5500|
| 65387| 5464|
|179736| 5439|
+------+-----+
only showing top 10 rows
```

```
In [37]: # TODO: output top 10 users by number of answers to questions by distinct users
         no_of_answers_sdf.show(10)
```

```
+-------+-----+
|   user|count|
+-------+-----+
|  22656|32020|
|1144035|25146|
|  29407|20836|
| 548225|16939|
| 157882|16609|
| 115145|16503|
|  17034|15436|
| 100297|15017|
|   6309|14276|
|  34397|14013|
+-------+-----+
only showing top 10 rows
```

1.7 Step 2.4.2 Results

```
In [38]: # TODO: number of users whose questions have never been answered or commented on
         len(list_op)
```

```
Out[38]: 150681
```

1.8 Step 2.4.3 Results

```
In [39]: # TODO: top 10 pairs of users by mutual answers, along with the number of questions the
         anscount_df = answers_sdf.groupby('from_node' , 'to_node').count()
         anscount_df1 = anscount_df.select(anscount_df['from_node'].alias("from_node_1"),
                                           anscount_df['to_node'].alias("to_node_1"),anscount_df

         anscount_df2 =  anscount_df.select(anscount_df['from_node'].alias("from_node_2"),
                                           anscount_df['to_node'].alias("to_node_2"),anscount_df

         concat_df = anscount_df1.join(anscount_df2 , ((anscount_df1['from_node_1'] ) == (anscou
                           ((anscount_df1['to_node_1']) == (anscount_df2['from_node_2']) ))
         ans_count = concat_df['count_1'] + concat_df['count_2']
         df = concat_df.withColumn("anscount" , ans_count)
         df = df[df.from_node_1 != df.to_node_1]

         df = df.select(df['from_node_1'].alias("user1"),
                                           df['to_node_1'].alias("user2"),df['anscount'].alias("

         df = df.sort('anscount' , ascending=False)
         df.dropDuplicates()
         df.show(10)
```

```
+-------+-------+--------+
|  user1|  user2|anscount|
+-------+-------+--------+
| 366797|  15168|      65|
|  15168| 366797|      65|
| 650492| 505088|      57|
| 505088| 650492|      57|
|1525840|1675891|      49|
|1675891|1525840|      49|
|2313718|2138752|      46|
|2138752|2313718|      46|
|1931641|1642617|      41|
|1642617|1931641|      41|
+-------+-------+--------+
only showing top 10 rows
```

2 Step 3

```
In [40]: # TODO: remove these, which just create dummy data
         #highest_indegree_sdf = spark.createDataFrame([{'node': 123, 'indegree': 4}])
         #highest_outdegree_sdf = spark.createDataFrame([{'node': 123, 'outdegree': 5}])

         # TODO: Fill in according to HW spec
         out_df = answers_sdf.groupby('from_node').count()
         out_df = out_df.select(out_df['from_node'].alias("node"),
                                 out_df['count'].alias("outdegree"))
out_df = out_df.sort('count' , ascending=False)
highest_outdegree_sdf=out_df

in_df  = answers_sdf.groupby('to_node').count()
in_df = in_df.select(in_df['to_node'].alias("node"),
                                 in_df['count'].alias("indegree"))

in_df = in_df.sort('count' , ascending=False)
highest_indegree_sdf=in_df
```

2.1 Step 3 Results

```
In [41]: highest_indegree_sdf.show(5)
```

```
+------+--------+
|  node|indegree|
+------+--------+
|  4653|    5453|
| 39677|    4971|
| 34537|    4391|
|179736|    3716|
| 84201|    3710|
+------+--------+
only showing top 5 rows
```

```
In [42]: highest_outdegree_sdf.show(5)
```

```
+-------+---------+
|   node|outdegree|
+-------+---------+
|  22656|    32030|
|1144035|    25146|
|  29407|    20842|
| 548225|    16944|
| 157882|    16615|
+-------+---------+
only showing top 5 rows
```

2.2 Step 4

```
In [1]: # TODO: insert code as you like
```

```
In [66]: #
         # Step 4.1 Pre-processing
```

```
        #
        def spark_bfs(G, origins, max_depth):
            ##Your logic goes here
            schema = StructType([StructField("node", IntegerType(), True)])
            my_sdf = spark.createDataFrame(origins, schema)
            frontier = my_sdf
            visited = spark.createDataFrame([],frontier.schema)
            for i in range(max_depth+1):
                if i == 0:
                    return_sdf = frontier
                    return_sdf = return_sdf.withColumn('new_col', F.lit(i)).alias('depth')
                else:
                    d1 = frontier.alias('f').join(G.alias('g'), F.col('f.node') == F.col('g.fro
                    d2 = d1.select(d1['to_node'].alias("node"))
                    visited = visited.unionAll(frontier)
                    frontier = d2
                    frontier = frontier.join(visited , frontier.node == visited.node , 'leftant
                    G = G.join(visited, G.to_node == visited.node, 'leftanti')
                    temp_df = frontier
                    temp_df = temp_df.withColumn('new_col', F.lit(i)).alias('depth')
                    return_sdf = return_sdf.unionAll(temp_df)

            return return_sdf
```

2.3 Step 4.1

```
In [67]: # TODO: comment out this line once your code is ready
         #bfs_sdf = spark.createDataFrame([{'node': 123, 'depth': 1}, {'node': 456, 'depth': 2}]
         # TODO: enable this once your code is ready
         origin_map = [{'node': 4550}, {'node': 242}]
         bfs_sdf = spark_bfs(comments_questions_sdf, origin_map, 2)
         bfs_sdf.cache()
         bfs_sdf.count()

Out[67]: 397
```

```
In [2]: # TODO: insert code as you like
```

2.4 Step 4.1 Results

```
In [68]: bfs_sdf.show(10)
```

```
+-------+-------+
|   node|new_col|
+-------+-------+
|   4550|      0|
|    242|      0|
|1619254|      1|
|2332659|      1|
|5504881|      1|
|1139389|      1|
|1940564|      1|
| 818089|      1|
|3047450|      1|
|4773326|      1|
+-------+-------+
only showing top 10 rows
```

2.5 Step 4.2

```
In [71]: #
         # Step 4.2 Pre-processing
         def create_filtered_bfs_sdf(input_sdf):
             filtered_bfs_sdf = input_sdf[input_sdf.new_col == 2]
             filtered_bfs_sdf = filtered_bfs_sdf.groupBy('node').count()
             filtered_bfs_sdf = filtered_bfs_sdf[filtered_bfs_sdf['count'] > 1]
             filtered_bfs_sdf.cache()
             return filtered_bfs_sdf

         def friend_rec(input_sdf, graph_sdf):
             G1=filtered_bfs_sdf.select(filtered_bfs_sdf['node'].alias("from_node"))
             G2=filtered_bfs_sdf.select(filtered_bfs_sdf['node'].alias("to_node"))
             G3=G1.join(G2, (G1['from_node'] < G2['to_node']))
             temp_sdf=graph_sdf.select(graph_sdf['from_node'],graph_sdf['to_node'])
             G3=G3.subtract(temp_sdf)
             temp_sdf=graph_sdf.select(graph_sdf['to_node'].alias('from_node'),graph_sdf['from_n
             friend_recommendations_sdf=G3.subtract(temp_sdf)
             return friend_recommendations_sdf
```

```
In [72]: # TODO: insert code as you like
```

```
In [73]: # TODO: comment this line out when your function works
         #friend_recommendations_sdf = spark.createDataFrame([\
                                                              # {'from_node': 123, 'to_node': 456}
                                                              #{'from_node': 456, 'to_node': 123}]
         # TODO: enable this when your function works
         filtered_bfs_sdf = create_filtered_bfs_sdf(bfs_sdf)
         friend_recommendations_sdf = friend_rec(filtered_bfs_sdf, comments_questions_sdf)
         friend_recommendations_sdf.cache()
         friend_recommendations_sdf.count()
```

```
Out[73]: 902
```

```
In [74]: friend_recommendations_sdf.show(5)
```

```
+---------+-------+
|from_node|to_node|
+---------+-------+
|    21918| 453447|
|    21918|4204628|
|    46646| 267679|
|    59017|1187554|
|   503032|2269511|
+---------+-------+
only showing top 5 rows
```

2.6 Step 4.2 Results

```
In [75]: friend_recommendations_sdf.show()
```

```
+---------+-------+
|from_node|to_node|
+---------+-------+
|    21918| 453447|
|    21918|4204628|
|    46646| 267679|
|    59017|1187554|
|   503032|2269511|
|    46646|1127460|
|    55503| 104015|
|    60602|1187554|
|   267679|1061543|
|  1218595|4800193|
|    27483|  46646|
|   104015|4595831|
|   211452|1894566|
|  1038179|1187554|
|    60602|  63775|
|   282194| 503032|
|   830423|4899760|
|    21918|  59017|
|    21918|1187554|
|    59017| 830423|
+---------+-------+
only showing top 20 rows
```

2.7 Step 4.3: Graph visualization

2.7.1 Once you have excecuted the cells in Step 4.2 and you have friend_recommendations_sdf, lets create friend_recommendations_df using toPandas(). This creates an in-memory dataFrame that we can use to build the graph. Here we have used ('from_node','to_node') as column names in friend_recommendations_sdf, please change it to what you have used in yours.

```
In [76]: import networkx as nx
         # TODO: create friend_graph NetworkX graph from friend_recommendations_df from friend_r
         n_sdf=friend_recommendations_sdf.toPandas()
         friend_graph=nx.from_pandas_dataframe(n_sdf,'from_node','to_node')
```

2.8 Step 4.3 Results

```
In [77]: print ("Number of nodes (characters) in this graph is", friend_graph.order()) # number
         print ("Number of edges in this graph is", len(friend_graph.edges())) # number of edges
         print ("Graph diameter is", nx.diameter(friend_graph)) # maximum eccentricity
```

```
Number of nodes (characters) in this graph is 43
Number of edges in this graph is 902
Graph diameter is 2
```

```
In [78]:  %matplotlib inline
          nx.draw(friend_graph)
```