Lab4_python_1

December 1, 2022

[12]:
```
import findspark
from pyspark import SparkConf
from pyspark import SparkContext

findspark.init()
spark = SparkContext.getOrCreate(SparkConf().setMaster("local[4]"))
spark = SparkSession(spark)
```

[13]:
```
from pyspark.sql.types import *
from graphframes import *
import pandas as pd
```

[14]:
```
fields = [
    StructField("id", StringType(), True),
    StructField("latitude", FloatType(), True),
    StructField("longitude", FloatType(), True),
    StructField("population", IntegerType(), True)
]
```

[15]:
```
v = spark.read.csv("/home/spark/lab04/task2/transport-nodes.csv", header=True,␣
↪schema=StructType(fields))
```

[16]:
```
src_dst = spark.read.csv("/home/spark/lab04/task2/transport-relationships.csv",␣
↪header=True)
```

[17]:
```
df_src_dst = src_dst.toPandas()
df_dst_src = src_dst.toPandas()
df_dst_src.columns = ["dst", "src", "relationship", "cost"]
```

[18]:
```
e = spark.createDataFrame(pd.concat([df_src_dst, df_dst_src], sort=False))
```

[20]:
```
g = GraphFrame(v, e)
```

[21]:
```
(g.vertices
 .filter("population > 100000 and population < 300000")
 .sort("population")
 .show())
```

```
+----------+--------+---------+----------+
|        id|latitude|longitude|population|
+----------+--------+---------+----------+
|Colchester|51.88921|  0.90421|    104390|
|   Ipswich|52.05917|  1.15545|    133384|
+----------+--------+---------+----------+
```

[22]:
```
from_expr = "id='Den Haag'"
to_expr = "population > 100000 and population < 300000 and id <> 'Den Haag'"
result = g.bfs(from_expr, to_expr)
```

[23]:
```
print(result.columns)
```

```
['from', 'e0', 'v1', 'e1', 'v2', 'e2', 'to']
```

[24]:
```
result = g.shortestPaths(["Colchester", "Immingham", "Hoek van Holland"])
result.sort(["id"]).select("id", "distances").show(truncate=False)
```

```
+---------------+------------------------------------------------------------+
|id             |distances                                                   |
+---------------+------------------------------------------------------------+
|Amsterdam      |[Immingham -> 1, Hoek van Holland -> 2, Colchester -> 4]|
|Colchester     |[Hoek van Holland -> 3, Immingham -> 3, Colchester -> 0]|
|Den Haag       |[Hoek van Holland -> 1, Immingham -> 2, Colchester -> 4]|
|Doncaster      |[Hoek van Holland -> 4, Immingham -> 1, Colchester -> 2]|
|Felixstowe     |[Immingham -> 4, Hoek van Holland -> 1, Colchester -> 2]|
|Gouda          |[Hoek van Holland -> 2, Immingham -> 3, Colchester -> 5]|
|Hoek van Holland|[Immingham -> 3, Hoek van Holland -> 0, Colchester -> 3]|
|Immingham      |[Hoek van Holland -> 3, Immingham -> 0, Colchester -> 3]|
|Ipswich        |[Immingham -> 4, Hoek van Holland -> 2, Colchester -> 1]|
|London         |[Hoek van Holland -> 4, Immingham -> 2, Colchester -> 1]|
|Rotterdam      |[Hoek van Holland -> 1, Immingham -> 3, Colchester -> 4]|
|Utrecht        |[Immingham -> 2, Hoek van Holland -> 3, Colchester -> 5]|
+---------------+------------------------------------------------------------+
```

[25]:
```
from graphframes.lib import AggregateMessages as AM
from pyspark.sql import functions as F
```

[30]:
```
add_path_udf = F.udf(lambda path, id: path + [id], ArrayType(StringType()))
```

[26]:
```
def shortest_path(g, origin, destination, column_name="cost"):
    if g.vertices.filter(g.vertices.id == destination).count() == 0:
        return (spark.createDataFrame(sc.emptyRDD(), g.vertices.schema) \
            .withColumn("path", F.array()))
    vertices = (g.vertices.withColumn("visited", F.lit(False))
```

```
        .withColumn("distance", F.when(g.vertices["id"] == origin, 0).
↪otherwise(float("inf")))        \
        .withColumn("path", F.array()))
    cached_vertices = AM.getCachedDataFrame(vertices)
    g2 = GraphFrame(cached_vertices, g.edges)
    while g2.vertices.filter('visited == False').first():
        current_node_id = g2.vertices.filter('visited == False').
↪sort("distance").first().id
        msg_distance = AM.edge[column_name] + AM.src['distance']
        msg_path = add_path_udf(AM.src["path"], AM.src["id"])
        msg_for_dst = F.when(AM.src['id'] == current_node_id,F.
↪struct(msg_distance, msg_path))
        new_distances = g2.aggregateMessages(F.min(AM.msg).
↪alias("aggMess"),sendToDst=msg_for_dst)
        new_visited_col = F.when(     \
            g2.vertices.visited | (g2.vertices.id == current_node_id),True).
↪otherwise(False)
        new_distance_col = F.when(new_distances["aggMess"].isNotNull() &    \
            (new_distances.aggMess["col1"] < g2.vertices.
↪distance),new_distances.aggMess["col1"])  \
            .otherwise(g2.vertices.distance)
        new_path_col = F.when(new_distances["aggMess"].isNotNull() &  \
            (new_distances.aggMess["col1"] < g2.vertices.distance),␣
↪new_distances.aggMess["col2"]   \
            .cast("array<string>")).otherwise(g2.vertices.path)
        new_vertices = (g2.vertices.join(new_distances,␣
↪on="id",how="left_outer")  \
            .drop(new_distances["id"])   \
            .withColumn("visited", new_visited_col)  \
            .withColumn("newDistance", new_distance_col)  \
            .withColumn("newPath", new_path_col)  \
            .drop("aggMess", "distance", "path")  \
            .withColumnRenamed('newDistance', 'distance')  \
            .withColumnRenamed('newPath', 'path'))
        cached_new_vertices = AM.getCachedDataFrame(new_vertices)
        g2 = GraphFrame(cached_new_vertices, g2.edges)
        if g2.vertices.filter(g2.vertices.id == destination).first().visited:
            return (g2.vertices.filter(g2.vertices.id == destination)  \
                .withColumn("newPath", add_path_udf("path", "id"))     \
                .drop("visited", "path")                               \
                .withColumnRenamed("newPath", "path"))
    return (spark.createDataFrame(sc.emptyRDD(), g.vertices.schema)    \
        .withColumn("path", F.array()))
```

[31]:
```
result = shortest_path(g, "Amsterdam", "Colchester", "cost")
result.select("id", "distance", "path").show(truncate=False)
```

```
+----------+--------+-----------------------------------------------------------
-------------+
|id        |distance|path
|
+----------+--------+-----------------------------------------------------------
-------------+
|Colchester|347.0   |[Amsterdam, Den Haag, Hoek van Holland, Felixstowe,
Ipswich, Colchester]|
+----------+--------+-----------------------------------------------------------
-------------+
```

[ ]: