Experiment 9: Page Rank Algorithm using PySpark

**Source Code:**

"""

This is an example implementation of PageRank. For more conventional use,

Please refer to PageRank implementation provided by graphx

Example Usage:

bin/spark-submit examples/src/main/python/pagerank.py data/mllib/pagerank\_data.txt 10

"""

from \_\_future\_\_ import print\_function

import os

os.environ["HADOOP\_HOME"] = "/home/hadoop/hadoop2"

os.environ["SPARK\_HOME"] = "/home/hadoop/spark-2.3.1-bin-hadoop2.7"

import findspark

findspark.init()

import re

import sys

from operator import add

from pyspark.sql import SparkSession

def computeContribs(urls, rank):

"""Calculates URL contributions to the rank of other URLs."""

num\_urls = len(urls)

for url in urls:

yield (url, rank / num\_urls)

def parseNeighbors(urls):

"""Parses a urls pair string into urls pair."""

parts = re.split(r'\s+', urls)

return parts[0], parts[1]

if \_\_name\_\_ == "\_\_main\_\_":

if len(sys.argv) != 3:

print("Usage: pagerank <file> <iterations>", file=sys.stderr)

sys.exit(-1)

# print("WARN: This is a naive implementation of PageRank and is given as an example!\n" +

# "Please refer to PageRank implementation provided by graphx",

# file=sys.stderr)

# Initialize the spark context.

spark = SparkSession\

.builder\

.appName("PythonPageRank")\

.getOrCreate()

# Loads in input file. It should be in format of:

# URL neighbor URL

# URL neighbor URL

# URL neighbor URL

# ...

lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])

# Loads all URLs from input file and initialize their neighbors.

links = lines.map(lambda urls: parseNeighbors(urls)).distinct().groupByKey().cache()

# print(links.groupByKey().map(lambda x : (x[0], list(x[1]))).collect())

print(links.count())

n = links.count()

# links.foreach(print)

# Loads all URLs with other URL(s) link to from input file and initialize ranks of them to one.

ranks = links.map(lambda url\_neighbors: (url\_neighbors[0], 1))

# Calculates and updates URL ranks continuously using PageRank algorithm.

for iteration in range(int(sys.argv[2])):

# Calculates URL contributions to the rank of other URLs.

contribs = links.join(ranks).flatMap(

lambda url\_urls\_rank: computeContribs(url\_urls\_rank[1][0], url\_urls\_rank[1][1]))

# Re-calculates URL ranks based on neighbor contributions.

ranks = contribs.reduceByKey(add).mapValues(lambda rank: rank \* 0.85 + 0.15)

# Collects all URL ranks and dump them to console.

for (link, rank) in ranks.collect():

print("%s has rank: %s." % (link, rank))

spark.stop()

**test.txt**

A B

C A

A C

B C

**Output:**

