PRINCIPLES OF BIG DATA MANAGEMENT

WARM UP PROJECT

Group – 10

1. Akhil Garidepally – ag8yx@umkc.edu
2. Gnathi Jonnalagadda -gjhnd@umsystem.edu
3. Naveen Krishna Koti – nkxbw@umsystem.edu
4. Sai Teja Polishetti - sp6n6@umkc.edu
5. Shireesha Maddi - smy44@umsystem.edu
6. Sujana Penmesta - sp3d3@umkc.edu

GIT Hub Link:

https://github.com/shireesha27/Big-data/blob/main/Warm_up_Project_Final.ipynb

Steps:

1. Create Google Colab account
2. Install JAVA, Apache Spark, Pyspark
3. Download the txt file and upload it in Google Colab
4. Import all the required libraries.
5. Read the text file using inbuilt methods.

```
[2]  # install java
     !apt-get install openjdk-8-jdk-headless -qq > /dev/null

     # install spark (change the version number if needed)
     !wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

     # unzip the spark file to the current folder
     !tar xf spark-3.0.0-bin-hadoop3.2.tgz

     # set your spark folder to your system path environment.
     import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"


     # install findspark using pip
     !pip install -q findspark
     #Importing Libraries
     import findspark
     findspark.init()
     findspark.find()
     import pyspark
     findspark.find()

     '/content/spark-3.0.0-bin-hadoop3.2'
```

```
[3]  pyspark

     <module 'pyspark' from '/content/spark-3.0.0-bin-hadoop3.2/python/pyspark/__init__.py'>

[4]  !pip install pyspark
     pyspark

     Collecting pyspark
       Downloading pyspark-3.1.2.tar.gz (212.4 MB)
          |████████████████████████████████| 212.4 MB 63 kB/s
     Collecting py4j==0.10.9
       Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
          |████████████████████████████████| 198 kB 46.5 MB/s
     Building wheels for collected packages: pyspark
       Building wheel for pyspark (setup.py) ... done
       Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl size=212880768 sha256=d40c5511c2a877ad02c8ea00fccfa324b7b67d3aba5a83238548bb2a7e1dc111
       Stored in directory: /root/.cache/pip/wheels/a5/0a/c1/9561f6fecb759579a7d863dcd846daaa95f598744e71b02c77
     Successfully built pyspark
     Installing collected packages: py4j, pyspark
     Successfully installed py4j-0.10.9 pyspark-3.1.2
     WARNING: The following packages were previously imported in this runtime:
       [py4j,pyspark]
     You must restart the runtime in order to use newly installed versions.

     RESTART RUNTIME
     <module 'pyspark' from '/content/spark-3.0.0-bin-hadoop3.2/python/pyspark/__init__.py'>
```

Task 1:

Question: Build a word cloud for NY Times articles using Apache Hadoop or Spark. (either platform is fine)

- List top 100 words used in all articles

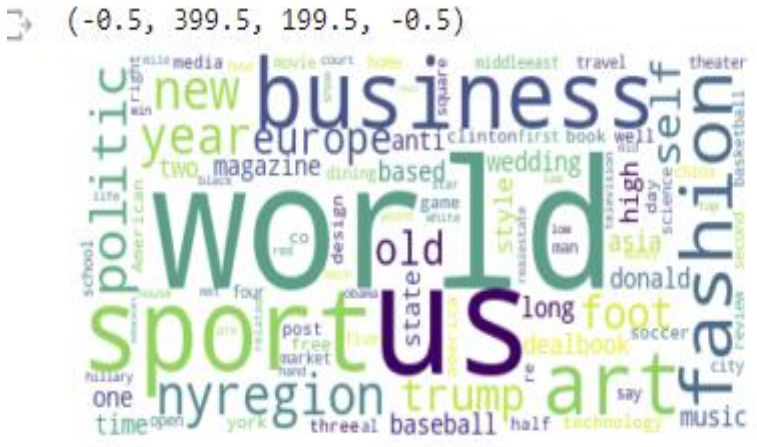
- Drop stop words [a, the, in, for, ….]

Steps for Task1:

- Imported all necessary libraries and loading the dataframe
- Created a spark schema (schema is the description of the structure of your data)
- Removing stop words from dataframe
- Creating a word cloud with max font size, top 100 number of words and lighten the background (Word cloud is a technique to show which word are the most frequent among the given text).
- Plotted the generated word cloud.

```
#task1
# Start with loading all necessary libraries
import matplotlib.pyplot as plt
from wordcloud import WordCloud,STOPWORDS
from pyspark.sql import SparkSession
from pyspark.sql.functions import  col
from pyspark.sql.types import *
spark = SparkSession.builder.appName("DataFrame").getOrCreate()

# Load in the dataframe
df1 = spark.sparkContext.textFile('/content/nytimes_news_articles.txt')

df1 = df1.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)
# Creating schema
schema = StructType([StructField("word", StringType(), True),
                     StructField("frequency", IntegerType(), True)])

df2 = spark.createDataFrame(df1, schema).orderBy(col("frequency"),ascending=False)

document = " ".join(row['word'] for row in df2.collect())

# Removing stopwords from dataframe
stop_words = ["html","nytimes"] + list(STOPWORDS)
# Creating word_cloud with text as argument in .generate() method
# Change the maximum number of word and lighten the backgrounds
wordcloud = WordCloud(collocations=False,stopwords = stop_words, max_words=100,background_color="white").generate(document)
# Display the generated Word Cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

Output:



Task 2:

Question: Also, the word cloud for top 5 news category

- Category that has the most of news articles
- URL contains the category which the news belong

For example:

- /us/politics/
- /world/
- /sports/
- /arts/ and so on

Steps for Task 2:

- Imported all necessary libraries and loading the dataframe
- Created a spark schema (schema is the description of the structure of your data)
- Applying filter to find most of news articles
- Creating a word cloud with max font size, top 5 news category and lighten the background (Word cloud is a technique to show which word are the most frequent among the given text).
- Plotted the generated word cloud.

```
#task2
# Start with loading all necessary libraries
import matplotlib.pyplot as plt
from wordcloud import WordCloud,STOPWORDS
from pyspark.sql import SparkSession
from pyspark.sql.functions import  col,split
from pyspark.sql.types import *
spark = SparkSession.builder.appName("DataFrame").getOrCreate()

# Load in the dataframe
df1 = spark.sparkContext.textFile('/content/nytimes_news_articles.txt')

df1 = df1.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)
# Creating schema
schema = StructType([StructField("word", StringType(), True),
                     StructField("frequency", IntegerType(), True)])

df2 = spark.createDataFrame(df1, schema)#.orderBy(col("frequency"))
df2 = df2.filter("word like '%http%'")
#print(df2.take(10))
df3 = split(df2["word"],"/")
df4 = df2.withColumn("word",df3.getItem(6)).select(col("word").cast("string")).fillna("")
#print(df4.take(10))
document = " ".join(row['word'] for row in df4.collect())
# Creating word_cloud with text as argument in .generate() method
# Removing stop words and Change the maximum number of word and lighten the backgrounds
wordcloud = WordCloud(collocations=False,stopwords = stop_words, background_color="white",max_words=5).generate(document)
# Display the generated Word Cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

Output:

Top 5 news category which are world, us, nyregion, sports and business

Task 3:

Question: List the top 10 words that are shared among the highest number of news articles in the same category

For example

- “student” appeared in 2443 articles under category “education”
- “trump” appeared in 457 articles under category “/us/politics”
- “baseball” appeared in 432 articles under category “sports”
- and so on.

Steps for Task 3:

- Imported all necessary libraries and loading the dataframe
- For applying for loop to find top 5 categories form categories and top 10 words that are shared among the highest number of news articles in the same category.
- Print the top 10 words from news categories

```
#TASK 3
from pyspark.sql import SparkSession
from pyspark.sql.functions import  col,split
from pyspark.sql.types import *
from collections import Counter
import findspark

"""Initilizing Spark Session"""
spark = SparkSession.builder \
    .master("local[*]") \
    .appName('warmupproject_task3') \
    .getOrCreate()
source = spark.sparkContext

"""Reading Input File"""
content = source.textFile("/content/nytimes_news_articles.txt")
emptyline_delete = content.filter(lambda x:len(x) > 0)
'''Deleting URl'''
url_delete = emptyline_delete.filter(lambda x:  not x.startswith("URL"))

"""STOP WORDS from Web"""
stopwords = ["-","a", "about", "above", "after","_", "again", "against", "don't","it's","ain", "all", "am", "an", "and", "any", "are", "aren", "aren't", "as", "at", "be", "because", "been", "be
             "eighty", "either", "else", "elsewhere", "end", "ending", "enough", "especially", "et", "etc", "even", "ever", "every", "everybody", "everyone", "everything", "everywhere", "ex", "
             "important", "inc", "indeed", "index", "information", "instead", "invention", "inward", "itd", "it'll", "j", "k", "keep", "keeps", "kept", "kg", "km", "know", "known", "knows", "l
             "wherein", "whereupon", "wherever", "whether","-", "which", "while", "whither", "who", "whoever", "whole", "whom", "whose", "why", "will", "with", "within", "without", "would", "ye

""" picking top 5 articles from text file"""
topfive_articles = {}
for y in content.collect():
    if y.startswith("URL"):
        article_type = y.split('/')[6]
        if article_type in topfive_articles:
            topfive_articles[article_type] += 1
        else:
            topfive_articles[article_type] = 1

topfive_category = list(dict(Counter(topfive_articles).most_common(5)).keys())
print(topfive_category)
current_url = ""
url_required = {}
```

```
for x in emptyline_delete.collect():
    if x.startswith("URL") :
        current_url = x
    else:
        if current_url in url_required:
            url_required[current_url] += x
        else:
            url_required[current_url] = x
"""retreving data based on the category"""
type_wise_data = {}
for category in topfive_category:
    for url in url_required:
        if category in url:
            if category in type_wise_data:
                type_wise_data[category] += url_required[url]
            else:
                type_wise_data[category] = url_required[url]

for a in topfive_category:
    topten_df = type_wise_data[a]
    df_words = topten_df.split(" ")

    df_words_replacing = [(x.lower()).replace(".", "").replace("'", "").replace(',', "").replace("“", "") for x in df_words]
    df_words_without_stopwords = [i for i in df_words_replacing if i not in stopwords]
    Counter(df_words_without_stopwords).most_common(10)
    # print(Counter(sports_words_without_stopwords).most_common(10))
    print("{} appeared in {} under category {}".format(Counter(df_words_without_stopwords).most_common(10)[0][0],Counter(df_words_without_stopwords).most_common(10)[0][1],a))
```

Output:

```
['sports', 'world', 'us', 'business', 'nyregion']
game appeared in 3589 under category sports
government appeared in 2279 under category world
trump appeared in 4933 under category us
percent appeared in 2670 under category business
york appeared in 1326 under category nyregion
```