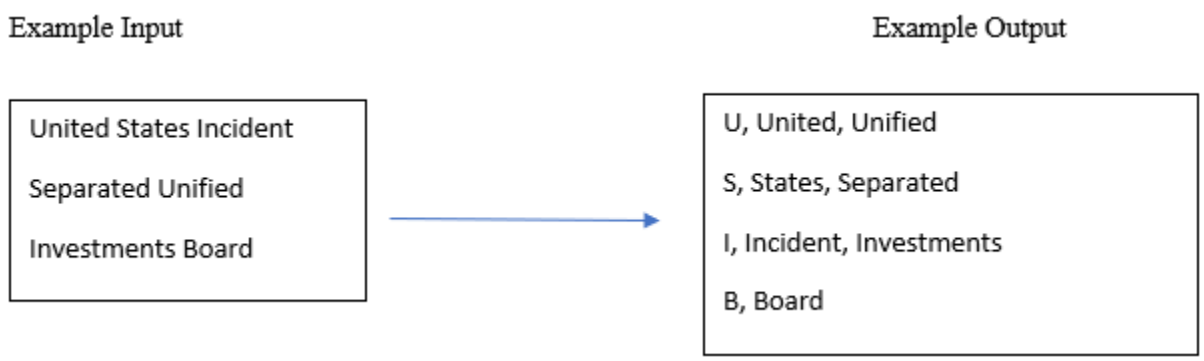
ICP 2

https://github.com/JAWolfe04/CS5542/wiki/ICP-2

Introduction

Write a spark program to group the words in a given text file based on their starting letters.
Use the Text File provided with ICP (icp2.txt)



ICP Requirements:

1. Spark Integration with Colab (or IDE that you are using) (50 points)
2. Creating a well commented Spark program and outputting the correct results and writing it to output file. (40 points) 3.Code quality, Pdf Report quality, video explanation (10 points)

Spark

Code
output.txt

The first part of the code involved downloading java, spark and pyspark:

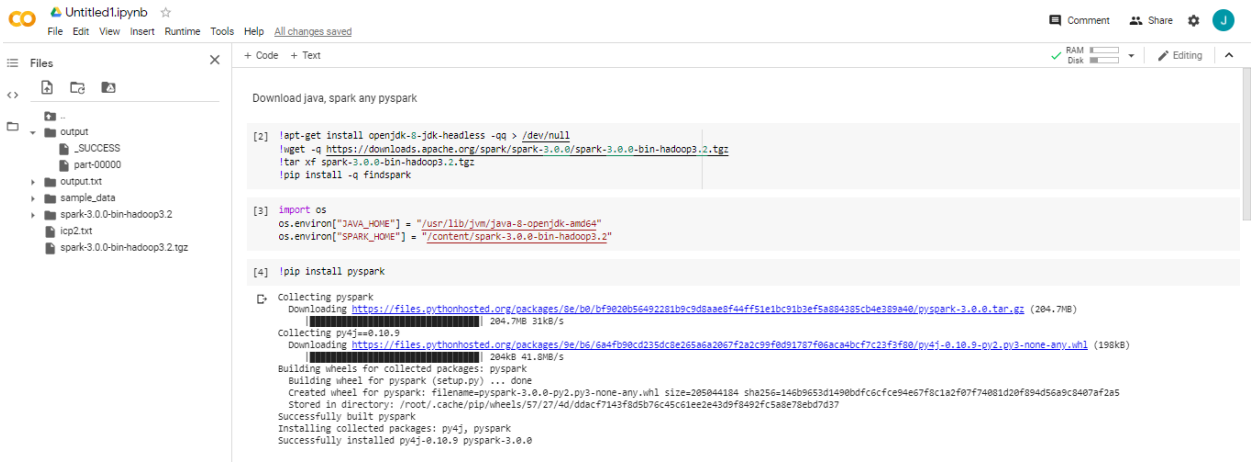
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz
!tar xf spark-3.0.0-bin-hadoop3.2.tgz
!pip install -q findspark

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"

!pip install pyspark
```

The following is a screenshot of the code executed in Google Colab:

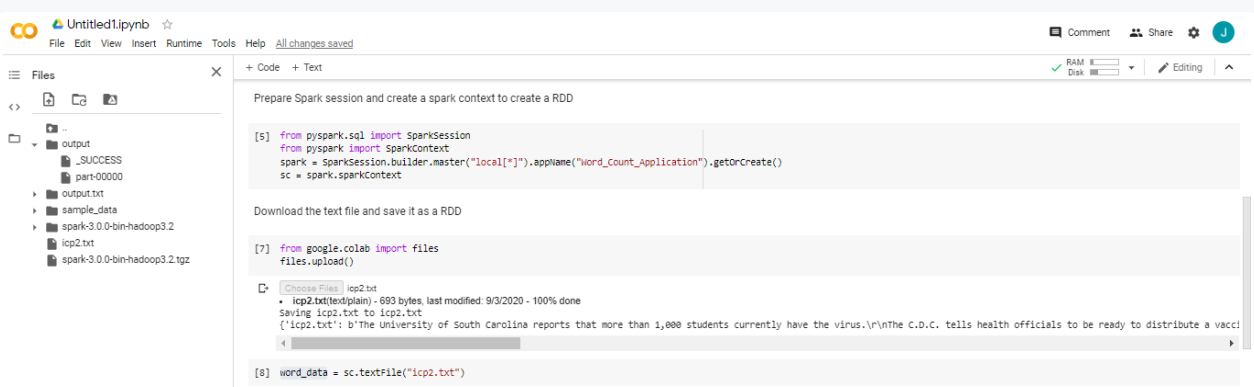


The next part involved preparing the Spark session, creating a spark context to create a RDD and importing the data into an RDD:

```
from pyspark.sql import SparkSession
from pyspark import SparkContext
spark =
SparkSession.builder.master("local[*]").appName("Word_Count_Application").getOrCreate
()
sc = spark.sparkContext

from google.colab import files
files.upload()

word_data = sc.textFile("icp2.txt")
```

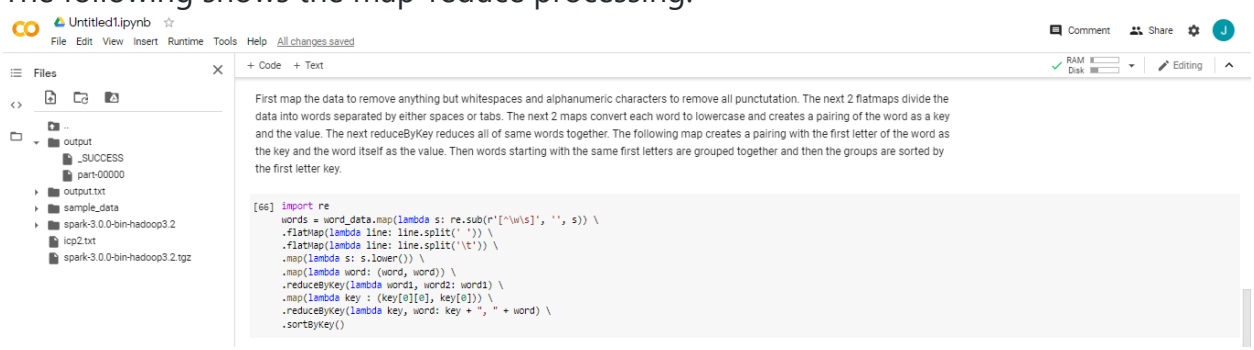


The next step performs all of the processing of the data using map-reduce. First map the data to remove anything but whitespaces and alphanumeric characters to remove all punctuation. The re library is imported to perform this regex processing. The next 2

flatmaps divide the data into words separated by either spaces or tabs. The next 2 maps convert each word to lowercase and creates a pairing of the word as a key and the value. The next reduceByKey reduces all of same words together. The following map creates a pairing with the first letter of the word as the key and the word itself as the value. Then words starting with the same first letters are grouped together and then the groups are sorted by the first letter key.

```
import re
words = word_data.map(lambda s: re.sub(r'[^\w\s]', '', s)) \
.flatMap(lambda line: line.split(' ')) \
.flatMap(lambda line: line.split('\t')) \
.map(lambda s: s.lower()) \
.map(lambda word: (word, word)) \
.reduceByKey(lambda word1, word2: word1) \
.map(lambda key : (key[0][0], key[0])) \
.reduceByKey(lambda key, word: key + ", " + word) \
.sortByKey()
```

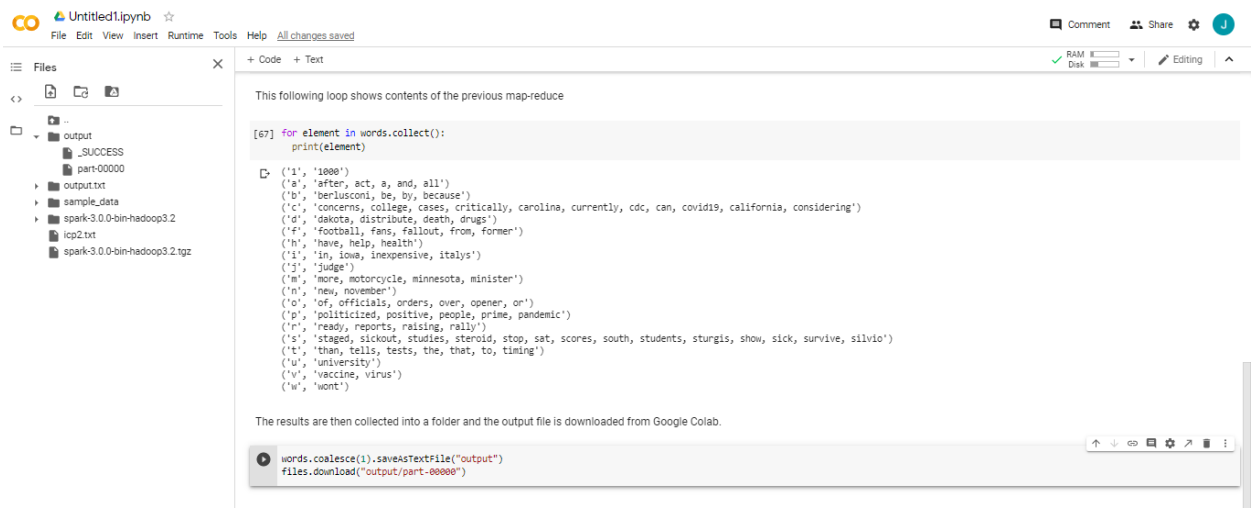
The following shows the map-reduce processing:



Finally, the results of the map-reduce are shown and output to a file:

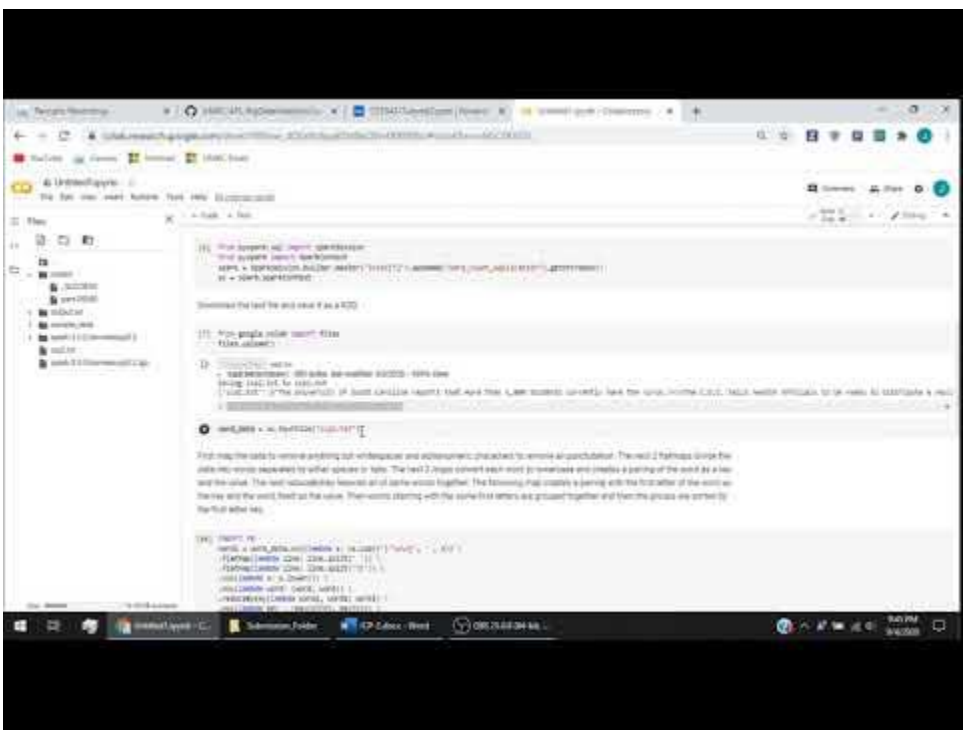
```
for element in words.collect():
  print(element)

words.coalesce(1).saveAsTextFile("output")
files.download("output/part-00000")
```



Youtube Video

https://www.youtube.com/watch?v=q-bki8LW1YA



References

No references were used in this ICP