Word Count

from mrjob.job import MRJob  
import re  
  
class MRWordCount(MRJob):  
  
 def mapper(self, \_, line):  
 words = re.findall(r'\b\w+\b', line)  
 for word in words:  
 yield word.lower(), 1  
  
 def reducer(self, word, counts):  
 yield word, sum(counts)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 MRWordCount.run()

txt file create it.

Bloom Filter

from pybloom\_live import BloomFilter  
  
capacity = 1000  
error\_rate = 0.01  
  
bloom\_filter = BloomFilter(capacity, error\_rate)  
  
bloom\_filter.add("apple")  
bloom\_filter.add("banana")  
bloom\_filter.add("orange")  
bloom\_filter.add("mango")  
  
print(" 'apple' in bloom\_filter:", "apple" in bloom\_filter)  
print(" 'grape' in bloom\_filter:", "grape" in bloom\_filter)  
print(" 'orange' in bloom\_filter:", "orange" in bloom\_filter)  
print(" 'mango' in bloom\_filter:", "mango" in bloom\_filter)  
  
bloom\_filter.tofile(open("bloom\_filter.bin", "wb"))  
  
loaded\_bloom\_filter = BloomFilter.fromfile(open("bloom\_filter.bin", "rb"))  
another\_bloom\_filter = BloomFilter(capacity, error\_rate)  
  
another\_bloom\_filter.add("grape")  
another\_bloom\_filter.add("kiwi")  
  
combined\_bloom\_filter = bloom\_filter.union(another\_bloom\_filter)  
  
print(" 'kiwi' in combined\_bloom\_filter:", "kiwi" in combined\_bloom\_filter)

Page Ranking

import numpy as np  
  
def pagerank(link\_matrix, damping\_factor=0.85, convergence\_threshold=1.0e-8, max\_iterations=100):  
 num\_pages = link\_matrix.shape[0]  
 page\_rank = np.ones(num\_pages) / num\_pages  
  
 for \_ in range(max\_iterations):  
 new\_page\_rank = damping\_factor \* np.dot(link\_matrix.T, page\_rank) + (1 - damping\_factor) / num\_pages  
  
 if np.linalg.norm(new\_page\_rank - page\_rank, 1) < convergence\_threshold:  
 return new\_page\_rank  
  
 page\_rank = new\_page\_rank  
 return page\_rank  
  
  
  
link\_matrix = np.array([  
 [0, 1, 0, 0],  
 [1, 0, 1, 1],  
 [1, 1, 0, 0],  
 [1, 0, 1, 0]  
], dtype=float)  
  
link\_matrix = link\_matrix / link\_matrix.sum(axis=0, keepdims=True)  
  
result = pagerank(link\_matrix)  
  
for i, page\_rank in enumerate(result):  
 print(f"Page {i+1}: {page\_rank}")

Apriori Algo.

import pandas as pd  
from mlxtend.preprocessing import TransactionEncoder  
from mlxtend.frequent\_patterns import apriori  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 transactions = [  
 ['bread', 'milk'],  
 ['bread', 'diaper', 'beer', 'egg'],  
 ['milk', 'diaper', 'beer', 'cola'],  
 ['bread', 'milk', 'diaper', 'beer'],  
 ['bread', 'milk', 'diaper', 'cola']  
 ]  
  
 te = TransactionEncoder()  
 te\_ary = te.fit(transactions).transform(transactions)  
 df = pd.DataFrame(te\_ary, columns=te.columns\_)  
  
 frequent\_itemsets = apriori(df, min\_support=0.5, use\_colnames=True)  
  
 print(frequent\_itemsets)

**Hadoop Commands**

Here is the list of several Hadoop commands:

**ls**

The "ls" command in Hadoop is employed to display a list of files, while the "lsr" option can be used for a recursive approach, providing a hierarchical view of a folder.

Syntax:

bin/hdfs dfs -ls <path> Example:

bin/hdfs dfs -ls /

It will print all the directories present in HDFS. bin directory contains executables so, bin/hdfs means we want the executables of hdfs particularly dfs(Distributed File System) commands.

**mkdir**

To create a directory. In Hadoop dfs there is no home directory by default. So let’s first create it.

Syntax:

bin/hdfs dfs -mkdir <folder name> creating home directory: hdfs/bin -mkdir /user

hdfs/bin -mkdir /user/username-> write the username of your computerExample:

bin/hdfs dfs -mkdir /geeks => '/' means absolute path

bin/hdfs dfs -mkdir geeks2 => Relative path -> the folder will be created relativeto the home directory.

**touchz**

The "touchz" command in Hadoop is used to create an empty file at the specified path in Hadoop Distributed File System (HDFS).

Syntax:

bin/hdfs dfs -touchz <file\_path> Example:

bin/hdfs dfs -touchz /geeks/myfile.txt

**copyFromLocal (or) put**

To copy files/folders from local file system to hdfs store. This is the most important command. Local filesystem means the files present on the OS.

Syntax:

bin/hdfs dfs -copyFromLocal <localfile path> <dest(present on hdfs)>

Example: Let’s suppose we have a file AI.txt on Desktop which we want to copy to folder geeks present on hdfs.

bin/hdfs dfs -copyFromLocal ../Desktop/AI.txt /geeks (OR)

bin/hdfs dfs -put ../Desktop/AI.txt /geeks

**cat**

To print file contents. Syntax:

bin/hdfs dfs -cat <path> Example:

// print the content of AI.txt present // inside geeks folder.

bin/hdfs dfs -cat /geeks/AI.txt ->

**copyToLocal (or) get**

To copy files/folders from hdfs store to local file system. Syntax:

bin/hdfs dfs -copyToLocal <<srcfile(on hdfs)> <local file dest> Example:

bin/hdfs dfs -copyToLocal /geeks (OR)

../Desktop/hero

bin/hdfs dfs -get /geeks/myfile.txt ../Desktop/hero myfile.txt from geeks folder will be copied to folder hero present on Desktop.moveFromLocal

This command will move file from local to hdfs. Syntax:

bin/hdfs dfs -moveFromLocal <local src> Example:

<dest(on hdfs)>

bin/hdfs dfs -moveFromLocal ../Desktop/cutAndPaste.txt

/geeks

**cp**

This command is used to copy files within hdfs. Let’s copy folder geeks to geeks\_copied. Syntax:

bin/hdfs dfs -cp <src(on hdfs)> <dest(on hdfs)> Example:

bin/hdfs -cp /geeks /geeks\_copied

**mv**

This command is used to move files within hdfs. Lets cut-paste a file myfile.txt from geeks folder to geeks\_copied.

Syntax:

bin/hdfs dfs -mv <src(on hdfs)> <src(on hdfs)> Example:

bin/hdfs -mv /geeks/myfile.txt /geeks\_copied

**rmr**

This command deletes a file from HDFS recursively. It is very useful command when you want to delete a non-empty directory.

Syntax:

bin/hdfs dfs -rmr <filename/directoryName> Example:

bin/hdfs dfs -rmr /geeks\_copied-> It will delete all the content inside thedirectory then the directory itself.

**du**

It will give the size of each file in directory. Syntax:

bin/hdfs dfs -du <dirName> Example:

bin/hdfs dfs -du /geeks