PROJECT 2

Department of Computer Science and Electrical Engineering

COMP-SCI 5540

Principles of Big Data Management

By TEAM 22

Manasa Thipparthi(mthwf)

Sai Jyothi Gudibandi(sg583)

Satya Harish Gumpalli(sgtm2)

Sri Harsha Kumar Raja Golla(sgnt8)

Project 2

Project Goal:

Develop a system to analyze and get the unique and duplicate words in Twitter's data using Hadoop MapReduce Framework.

Tasks Included:

- MapReduce program to find the list of words that have duplicates in the tweets' text.
- MapReduce program to find the list of words that are unique in the tweets' text.
- Store the lists in two text files: dups.txt and uniqs.txt
- Print the ratio of the number of unique words to the number of words with duplicates.
- Extra Requirement:
    - Implement a MapReduce program to determine the best time to post a tweet.
        - Propose the metric/criterion of your choice based on the tweet JSON format.
        - Run your program and return the top ten best times to post a tweet on twitter.

Prerequisite Skills:

- Create, open, read, and write files using a local file system.
- Write a basic word count function in MapReduce.
- Read and parse a JSON file. Perform a word count on one attribute on a list of JSON objects.

Our Project Plan:

We have collected the tweets on movies and implemented Map/Reduce programs to determine the vocabulary uniqueness of our dataset.

- **Module 1:**
  Our initial step was to find out the list of words that have duplicates in the tweets text. We accomplished this task by using Map Reduce concept. We have written a Map Reduce program that reads the tweets file and then finds the duplicate words in the tweets text.

- **Module 2:**
  In this module, we have written a python program using MapReduce concept. By using this program we got the list of unique words in the tweets data.

- **Module 3:**
  In this phase, we have stored the outputs of the module1 and module2 i.e. list of duplicate words and the list of unique words in tweets text as two text files: dups.txt and uniqs.txt respectively.

- **Module 4:**
  Here by using MapReduce program we have retrieved the count of unique words and count of duplicate words and calculated the ratio of number of unique words to the number of duplicate words.

Extra Requirement:

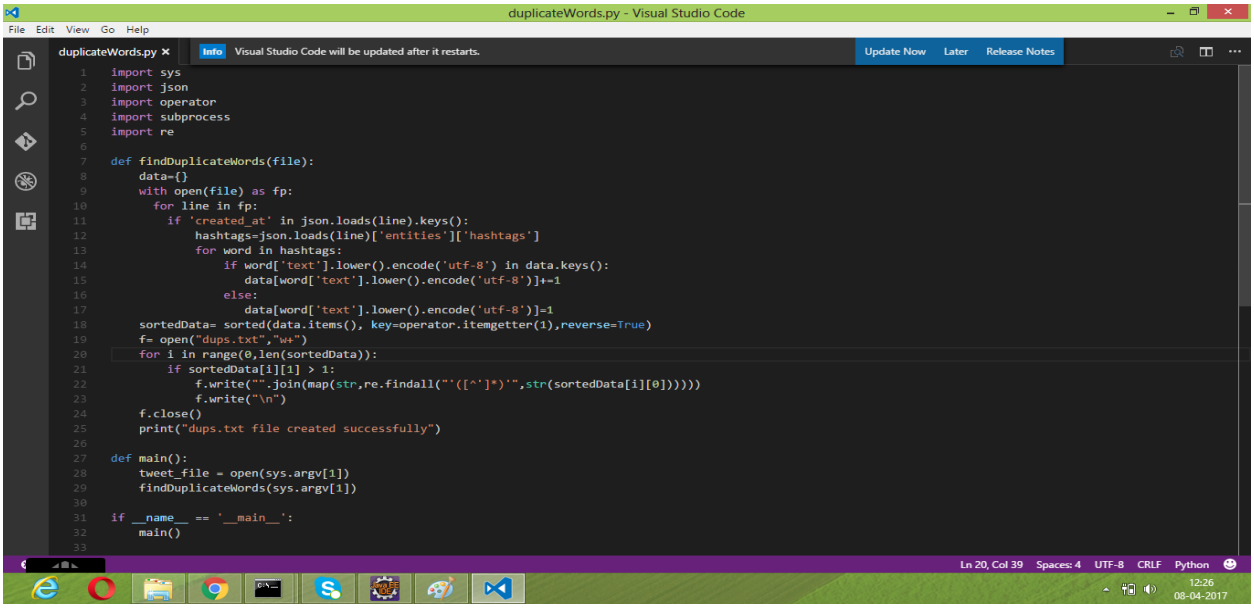
- In part of Extra Requirement we have implemented a MapReduce program to determine the best time to post a tweet.
  - **Metric:** We have considered one hour of tweets and analyzed the number of tweets for every minute in that one hour. Based on the number of tweets per minute, the minute which had highest number of tweets was considered as the best time to post a tweet.
  - And we have implemented a program that will give output as the top ten best times to post a tweet on twitter.

Below are the screenshots of corresponding module outputs:

Module 1:

MapReduce program to find the list of words that have duplicates in the tweets' text.

Program & Execution:



Module 2:

MapReduce program to find the list of words that have uniques in the tweets' text.

Program & Execution:

uniqueWords.py - Visual Studio Code

```python
import sys
import json
import operator
import subprocess
import re

def findUniqueWords(file):
    data={}
    with open(file) as fp:
      for line in fp:
        if 'created_at' in json.loads(line).keys():
            hashtags=json.loads(line)['entities']['hashtags']
            for word in hashtags:
                if word['text'].lower().encode('utf-8') in data.keys():
                   data[word['text'].lower().encode('utf-8')]+=1;
                else:
                   data[word['text'].lower().encode('utf-8')]=1
    sortedData= sorted(data.items(), key=operator.itemgetter(1),reverse=True)
    f= open("uniqs.txt","w+")
    for i in range(0,len(sortedData)):
        if sortedData[i][1] == 1:
            f.write("".join(map(str,re.findall("'([^']*)'",str(sortedData[i][0])))));
            f.write("\n");
    f.close();
    print("uniqs.txt file created successfully")

def main():
    tweet_file = open(sys.argv[1])
    findUniqueWords(sys.argv[1])

if __name__ == '__main__':
    main()
```

C:\Windows\system32\cmd.exe

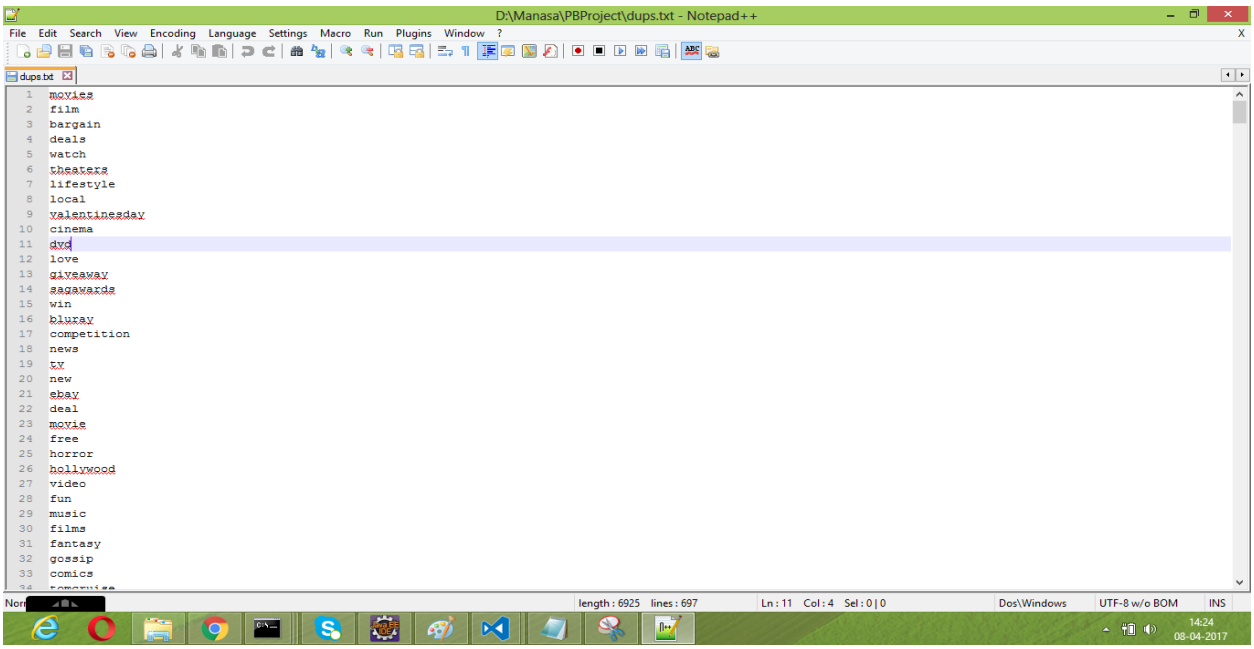
```
D:\Manasa\PBProject>python uniqueWords.py tweets.json
uniqs.txt file created successfully

D:\Manasa\PBProject>
```

Module 3:

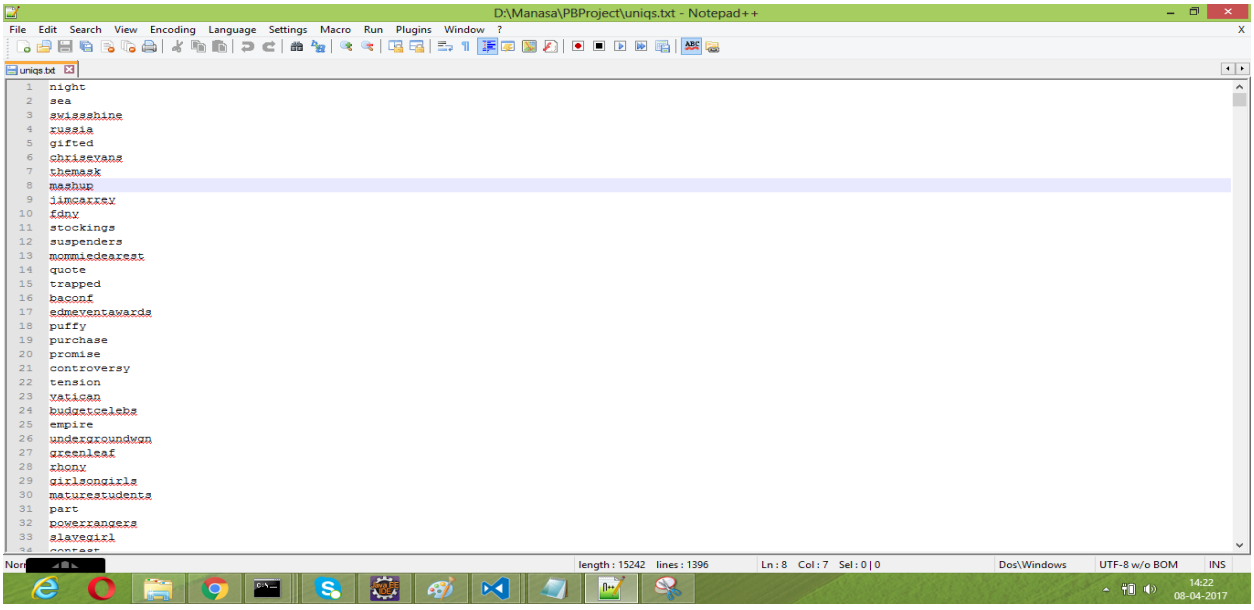
Storing the results of module1 and module2 lists in two text files: dups.txt and uniqs.txt

Output :

dups.txt:



uniqs.txt:



Module 4:

Printing the ratio of the number of unique words to the number of words with duplicates.

Program:

ratioofUniquetoDuplicate.py - Visual Studio Code

```python

def findRatoofUniquetoDuplicate(file):
    data={}
    uniqueWordsCount = 0
    duplicateWordsCount = 0
    ratioofUniquetoDuplicate = 0
    with open(file) as fp:
      for line in fp:
        if 'created_at' in json.loads(line).keys():
            hashtags=json.loads(line)['entities']['hashtags']
            for word in hashtags:
                if word['text'].lower().encode('utf-8') in data.keys():
                   data[word['text'].lower().encode('utf-8')]+=1;
                else:
                   data[word['text'].lower().encode('utf-8')]=1
    sortedData = sorted(data.items(), key=operator.itemgetter(1),reverse=True)
    for i in range(0,len(sortedData)):
        if sortedData[i][1] > 1:
            duplicateWordsCount = duplicateWordsCount + 1
        elif sortedData[i][1] == 1:
            uniqueWordsCount = uniqueWordsCount + 1
    print("Unique words count : ",uniqueWordsCount)
    print("Duplicate Words Count : ",duplicateWordsCount)
    ratio = uniqueWordsCount/duplicateWordsCount
    print("Ratio of number of Unique words to number of words with duplicates : ",int(ratio))

def main():
    tweet_file = open(sys.argv[1])
    findRatoofUniquetoDuplicate(sys.argv[1])

if __name__ == '__main__':
    main()
```

Output:

C:\Windows\system32\cmd.exe

```
D:\Manasa\PBProject>python ratioofUniquetoDuplicate.py tweets.json
Unique words count :  1395
Duplicate Words Count :  696
Ratio of number of Unique words to number of words with duplicates -
Unique Words : Duplicate Words =  2  : 1

D:\Manasa\PBProject>
```

Extra Requirement:

MapReduce program to determine the best time to post a tweet.

Program:

bestTimetoTweet.py - Visual Studio Code

```python
import sys
import json
import operator
import subprocess
import re
from datetime import datetime

def calculateBestTime(file):
    data={}
    with open(file) as fp:
      for line in fp:
        if 'created_at' in json.loads(line).keys():
            createdAt=json.loads(line)['created_at']
            date= datetime.strptime(createdAt, "%a %b %d %H:%M:%S +0000 %Y")
            hour = date.hour
            if date.minute in data.keys():
                data[date.minute]+=1;
            else:
                data[date.minute]=1;
    sortedData=sorted(data.items(), key=operator.itemgetter(1),reverse=True)
    print("Best time to Tweet : ",hour,":",sortedData[0][0],"-",hour,":",sortedData[0][0]+1)
def main():
    tweet_file = open(sys.argv[1])
    calculateBestTime(sys.argv[1])

if __name__ == '__main__':
    main()
```

Output:

C:\Windows\system32\cmd.exe

```
D:\Manasa\PBProject>python bestTimetoTweet.py bestTimeSampleTweets.json
Best time to Tweet :  17 : 49 - 17 : 50

D:\Manasa\PBProject>
```

MapReduce program which will give output as the top ten best times to post a tweet on twitter.

Program:

topTenBestTimestoTweet.py - Visual Studio Code

```python
import sys
import json
import operator
import subprocess
import re
from datetime import datetime

def claculateBestTime(file):
    data={}
    with open(file) as fp:
      for line in fp:
        if 'created_at' in json.loads(line).keys():
            createdAt=json.loads(line)['created_at']
            date= datetime.strptime(createdAt, "%a %b %d %H:%M:%S +0000 %Y")
            hour= date.hour
            if date.minute in data.keys():
                data[date.minute]+=1;
            else:
                data[date.minute]=1;
    sortedData=sorted(data.items(), key=operator.itemgetter(1),reverse=True)
    print("Top 10 Best Times to Tweet :")
    for i in range(0,10):
        print(hour,":",sortedData[i][0],"-",hour,":",sortedData[i][0]+1)
def main():
    tweet_file = open(sys.argv[1])
    claculateBestTime(sys.argv[1])

if __name__ == '__main__':
    main()
```

Output:

C:\Windows\system32\cmd.exe

```
D:\Manasa\PBProject>python topTenBestTimestoTweet.py bestTimeSampleTweets.json
Top 10 Best Times to Tweet :
17 : 49 - 17 : 50
17 : 54 - 17 : 55
17 : 14 - 17 : 15
17 : 13 - 17 : 14
17 : 26 - 17 : 27
17 : 0 - 17 : 1
17 : 1 - 17 : 2
17 : 15 - 17 : 16
17 : 16 - 17 : 17
17 : 21 - 17 : 22

D:\Manasa\PBProject>
```

GitHub URL: https://github.com/ManasaReddyThipparthi/PBProject-TwitterAnalysis/tree/master/Project-2