CSE487 Proejct2 Report

Weiyang Chen
Zhiwen Huang

April 18th, 2019

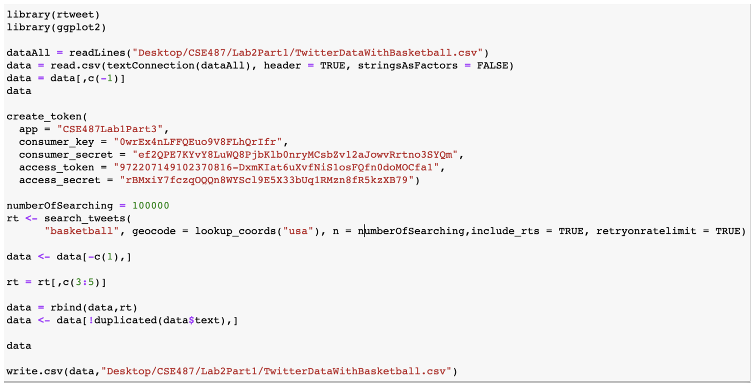
1 Introduction

The goal for his Lab is selecting a topic, and using Twitter API, New York Times API and Commcon Crwal to collect three different data about your topic. And using the hadoop to run the Mapper and Reducer to get each data's top 10 words with the highest appear frequency. After getting ten words. Change Mapper and reducer code that help to calculate Co-Occurrence foe each two words of ten. Then Using Tableau to draw the graph about their relationship and post in a web. In this report I will describe how to collect each data and how to use different tools to finish this project.

2 Part 1 : Prototype Data Collection

2.1 Twitter API Data Collection

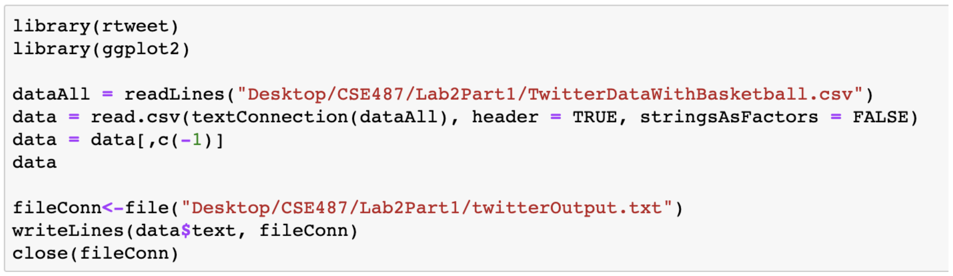
Step 1 : Using R language to run the Twitter API with a key word. And it will return us a set of data. Using write file function to store those data into a 'CSV' type data file.



```
library(rtweet)
library(ggplot2)

dataAll = readLines("Desktop/CSE487/Lab2Part1/TwitterDataWithBasketball.csv")
data = read.csv(textConnection(dataAll), header = TRUE, stringsAsFactors = FALSE)
data = data[,c(-1)]
data

create_token(
  app = "CSE487Lab1Part3",
  consumer_key = "0wrEx4nLFFQEuo9V8FLhQrIfr",
  consumer_secret = "ef2QPE7KYvY8LuWQ8PjbKlb0nryMCsbZv12aJowvRrtno3SYQm",
  access_token = "972207149102370816-DxmKIat6uXvfNiS1osFQfn0doMOCfa1",
  access_secret = "rBMxiY7fczqOQQn8WYScl9E5X33bUq1RMzn8fR5kzXB79")

numberOfSearching = 100000
rt <- search_tweets(
      "basketball", geocode = lookup_coords("usa"), n = numberOfSearching,include_rts = TRUE, retryonratelimit = TRUE)

data <- data[-c(1),]

rt = rt[,c(3:5)]

data = rbind(data,rt)
data <- data[!duplicated(data$text),]

data

write.csv(data,"Desktop/CSE487/Lab2Part1/TwitterDataWithBasketball.csv")
```

Figure 1: Twitter API Search and Store

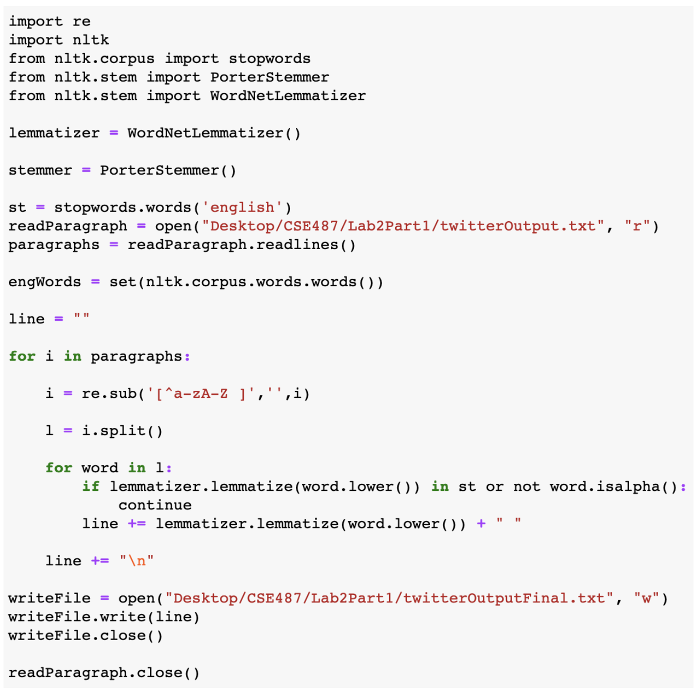
Step 2 : Using R language to read the twitter 'CSV' type file line by line (Each line will only contains one twitter message). And at same time add them into a txt type data file line by line. In the end, output this file.



```r
library(rtweet)
library(ggplot2)

dataAll = readLines("Desktop/CSE487/Lab2Part1/TwitterDataWithBasketball.csv")
data = read.csv(textConnection(dataAll), header = TRUE, stringsAsFactors = FALSE)
data = data[,c(-1)]
data

fileConn<-file("Desktop/CSE487/Lab2Part1/twitterOutput.txt")
writeLines(data$text, fileConn)
close(fileConn)
```

Figure 2: Change 'csv' file to 'txt' file

Step 3 : Using Python to read the output.txt file comes from step 2 output.Then using python's ntlk package to remove the stop word from the output.txt, and using re library to remove the special symbols. In the end, using WordNetLemmatizer to do Lemmatization.(ex. players - player) And store as finalOutput.txt file.(This file will be using in the P2 and P3 to run MR)

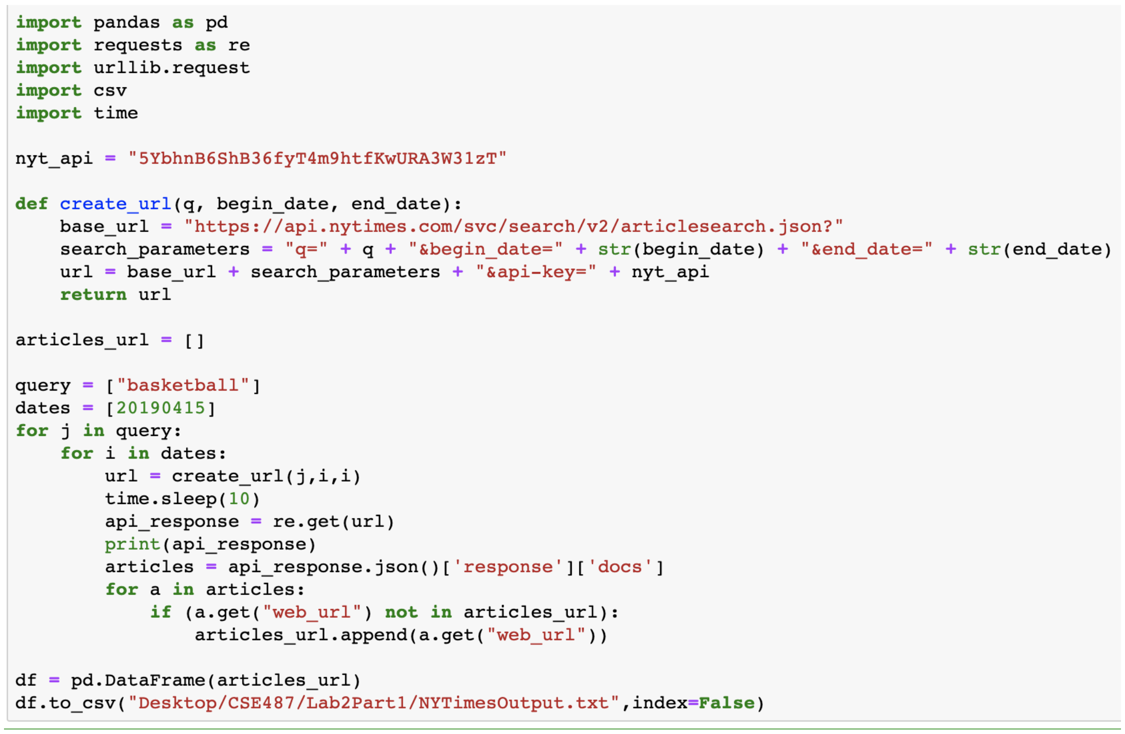


```python
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

stemmer = PorterStemmer()

st = stopwords.words('english')
readParagraph = open("Desktop/CSE487/Lab2Part1/twitterOutput.txt", "r")
paragraphs = readParagraph.readlines()

engWords = set(nltk.corpus.words.words())

line = ""

for i in paragraphs:

    i = re.sub('[^a-zA-Z ]','',i)

    l = i.split()

    for word in l:
        if lemmatizer.lemmatize(word.lower()) in st or not word.isalpha():
            continue
        line += lemmatizer.lemmatize(word.lower()) + " "

    line += "\n"

writeFile = open("Desktop/CSE487/Lab2Part1/twitterOutputFinal.txt", "w")
writeFile.write(line)
writeFile.close()

readParagraph.close()
```

Figure 3: Remove Stop Word and Special Symobols and lemmatization

2.2 New York Times API Data Collection

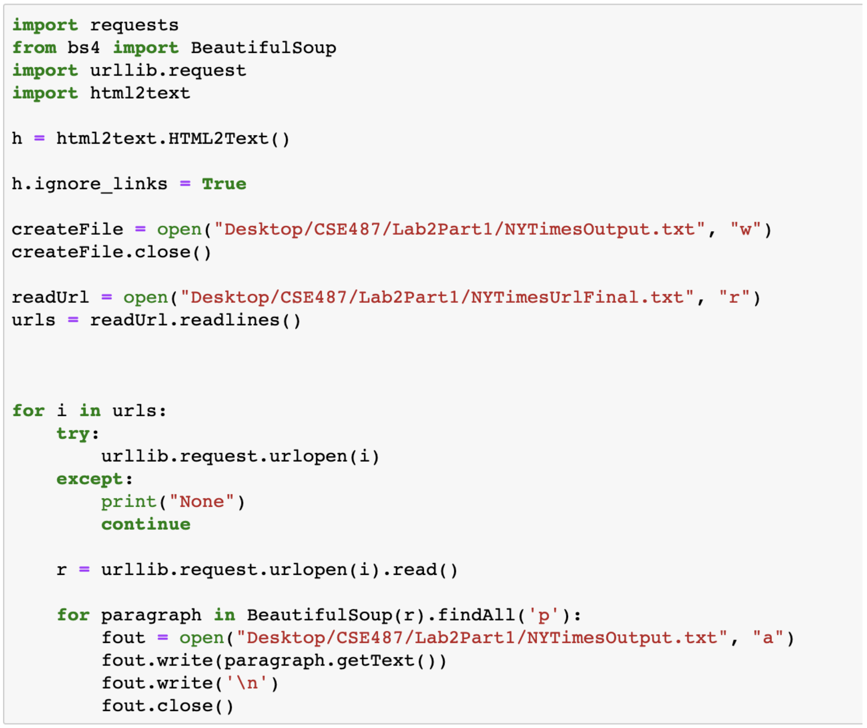
Step 1 : Using New York Times API with python to collect the urls of the articles about the topic. And store the urls into a txt file.



```python
import pandas as pd
import requests as re
import urllib.request
import csv
import time

nyt_api = "5YbhnB6ShB36fyT4m9htfKwURA3W31zT"

def create_url(q, begin_date, end_date):
    base_url = "https://api.nytimes.com/svc/search/v2/articlesearch.json?"
    search_parameters = "q=" + q + "&begin_date=" + str(begin_date) + "&end_date=" + str(end_date)
    url = base_url + search_parameters + "&api-key=" + nyt_api
    return url

articles_url = []

query = ["basketball"]
dates = [20190415]
for j in query:
    for i in dates:
        url = create_url(j,i,i)
        time.sleep(10)
        api_response = re.get(url)
        print(api_response)
        articles = api_response.json()['response']['docs']
        for a in articles:
            if (a.get("web_url") not in articles_url):
                articles_url.append(a.get("web_url"))

df = pd.DataFrame(articles_url)
df.to_csv("Desktop/CSE487/Lab2Part1/NYTimesOutput.txt",index=False)
```

Figure 4: Using New York API And Store URLs

Step 2 : Read the file which stores URLs, Using ’urllib’ and ’BeautifulSoup’ to scrap the website. ’Urllib’ can help us to access the website by its url. And by read() function to return the website’s html as string to us. And then using ’BeautifulSoup’ to read html string, and find the sentence with p attribute. Scrap out and add to the output.txt file. (Each line will contain only one sentence) So if we iterator a article we can get every sentence from this article and exclude unused information or Advertising. After iterator the all the URLs, return an output.txt file with articles’sentences.



```python
import requests
from bs4 import BeautifulSoup
import urllib.request
import html2text

h = html2text.HTML2Text()

h.ignore_links = True

createFile = open("Desktop/CSE487/Lab2Part1/NYTimesOutput.txt", "w")
createFile.close()

readUrl = open("Desktop/CSE487/Lab2Part1/NYTimesUrlFinal.txt", "r")
urls = readUrl.readlines()



for i in urls:
    try:
        urllib.request.urlopen(i)
    except:
        print("None")
        continue

    r = urllib.request.urlopen(i).read()

    for paragraph in BeautifulSoup(r).findAll('p'):
        fout = open("Desktop/CSE487/Lab2Part1/NYTimesOutput.txt", "a")
        fout.write(paragraph.getText())
        fout.write('\n')
        fout.close()
```

Figure 5: Access Website by URL and Scrap Article

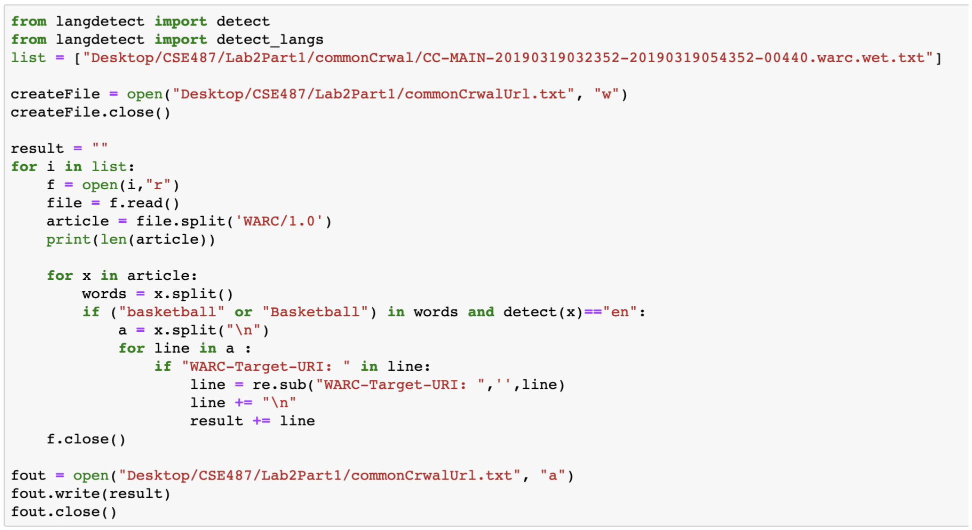
Step 3 : Remove Stop Words, Special Symbols, and Lemmatization. Same way to do as Step 3 in section 2.1 Twitter API Data Colleciton (Same Code With Figure 3)

2.3 Common Crwal Data Collection

Step 1 : Download the Commcon Crwal Wet File(Each wet file contains a huge number of random Articles with their URL), Change its type to .txt that is readable.

Figure 6: Download wet file and change to .txt

Step 2 : Using python to iterator whole file and filter out the articles about our topic. The Way I did is to read file and save into a string. Using split() function to split string into each article. Iterator each word for each article. If a article contains the word match the topic. Then Scrap the Articles' URL. In the store the URLs we scraped and store into a URL.txt file.



```
from langdetect import detect
from langdetect import detect_langs
list = ["Desktop/CSE487/Lab2Part1/commonCrwal/CC-MAIN-20190319032352-20190319054352-00440.warc.wet.txt"]

createFile = open("Desktop/CSE487/Lab2Part1/commonCrwalUrl.txt", "w")
createFile.close()

result = ""
for i in list:
    f = open(i,"r")
    file = f.read()
    article = file.split('WARC/1.0')
    print(len(article))

    for x in article:
        words = x.split()
        if ("basketball" or "Basketball") in words and detect(x)=="en":
            a = x.split("\n")
            for line in a :
                if "WARC-Target-URI: " in line:
                    line = re.sub("WARC-Target-URI: ",'',line)
                    line += "\n"
                    result += line
    f.close()

fout = open("Desktop/CSE487/Lab2Part1/commonCrwalUrl.txt", "a")
fout.write(result)
fout.close()
```

Figure 7: Get URL From common Crwal .wet File

Step 3 : Read the file which stores URLs and Scrap the p attribute content from each URL. Same way to do as Step 2 in the section 2.2 New York Times API Data Collection. (Same Code with Figure 5)

Step 4 : Remove Stop Words, Special Symbols, and Lemmatization. Same way to do as Step 3 in section 2.1 Twitter API Data Colleciton (Same Code With Figure 3)

3 Part 2 : Set Up Big Data Infrastructure

3.1 Install Docker

Go to docker installation page and select the appropriate OS that you are currently using and follow the instructions

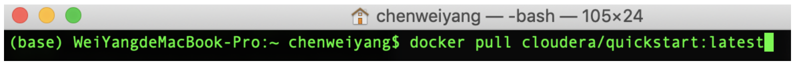
3.2 Increase RAM size for Docker

On Windows: On the right end of your task bar, right click on Docker image Settings Advanced and select the RAM size to 8GB. Close and Restart your system.

On Mac On the top of your menu bar you can check docker icon, click on that Preferences Advanced select the RAM size to 8GB.

3.3 Pull the Cloudera-Quickstart docker Image

Open the Terminal and Run ”docker pull cloudera/quickstart:last” to pull out nestest cloudera verison.



```
(base) WeiYangdeMacBook-Pro:~ chenweiyang$ docker pull cloudera/quickstart:latest
```

Figure 8: Pull Cloudera Images From Docker

3.4 Create Local Directory for Your MR solution and data

Create a local file name ”dockerMR” and add local data(Comes From P1) and Mapper and Reducer Program in to it.

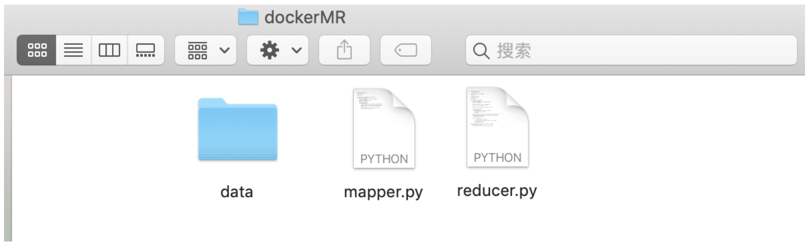
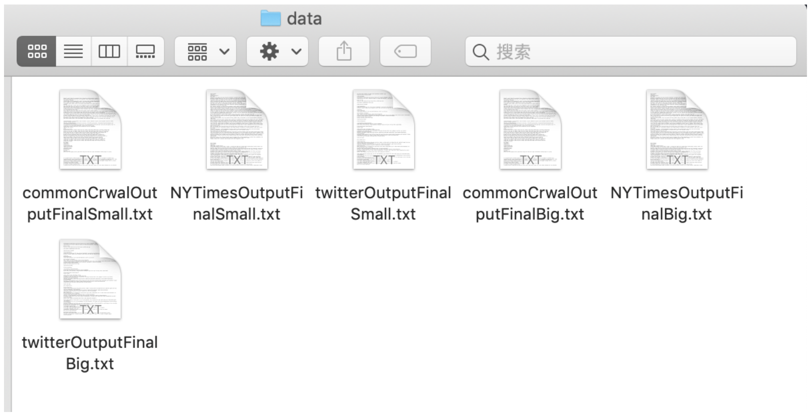


Figure 9: dockerMR File

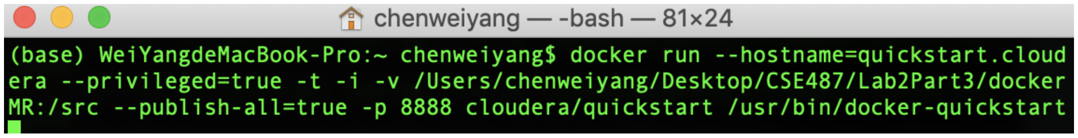


data
搜索
TXT
commonCrwalOutputFinalSmall.txt
TXT
NYTimesOutputFinalSmall.txt
TXT
twitterOutputFinalSmall.txt
TXT
commonCrwalOutputFinalBig.txt
TXT
NYTimesOutputFinalBig.txt
TXT
twitterOutputFinalBig.txt

Figure 10: Data in dockerMR

3.5 Configure docker dirctory and map to local workspace

Run the following command. ”docker run –hostname=quickstart.cloudera –privileged=true -t -i -v localpath:/src –publish-all=true -p 8888 cloudera/quickstart /usr/bin/docker-quickstart”. (replace localpath with a location you want to map)



```
(base) WeiYangdeMacBook-Pro:~ chenweiyang$ docker run --hostname=quickstart.cloud
era --privileged=true -t -i -v /Users/chenweiyang/Desktop/CSE487/Lab2Part3/docker
MR:/src --publish-all=true -p 8888 cloudera/quickstart /usr/bin/docker-quickstart
```

Figure 11: Command to run Cloudra in the Docker

3.6 Provision Data

Input folloing commands

hadoopfs − mkdir/user/yourName
hadoopfs − mkdir/user/yourName/MR
hadoopfs − mkdir/user/yourName/MR/input
cd/src/data/
hadoopfs − put ∗ .txt/user/yourName/MR/input
hadoopfs − ls/user/yourName/MR/input/
cd..

3.7 Process the Data using MR

Now Run the MapReduce program using the folloing command.
hadoopjar/usr/lib/hadoop−mapreduce/hadoop−streaming−2.6.0−cdh5.7.0.jar
−file/src/mapper.py − mapper/src/mapper.py
−file/src/reducer.py − reducer/src/reducer.py
−input/user/yourName/MR/input/ ∗ −output
/user/yourName/MR/output

3.8 Observe the output

Observe the output generated by MR and move it to your local file system for further processing.
hadoopfs − cat/user/bina/MR/output/part∗
hadoopfs−get/user/bina/MR/output//src/

4 Part 3 : Analyze and visualize

4.1 Execute the MapReduce Word Count

4.1.1 Run the MapReduce Program

Follow the instruction of part 2 to using docker run the cloudera-quick start with your mapreducer program. After the program run success, you will get three output file (Twitter, commcon crwal, New York Times) with word count result by MR program.

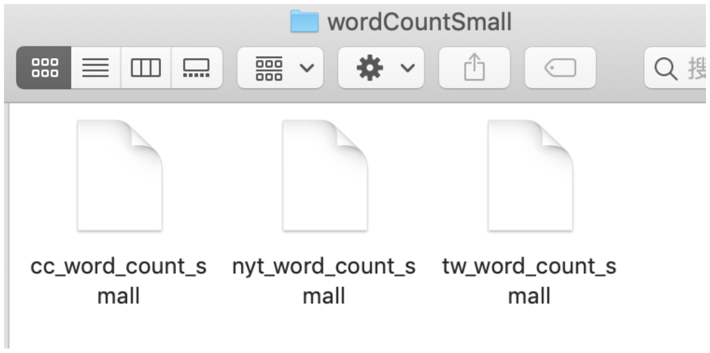


Figure 12: Result After Runing MR

4.1.2 Stemming and removing stop words And Output A CSV File

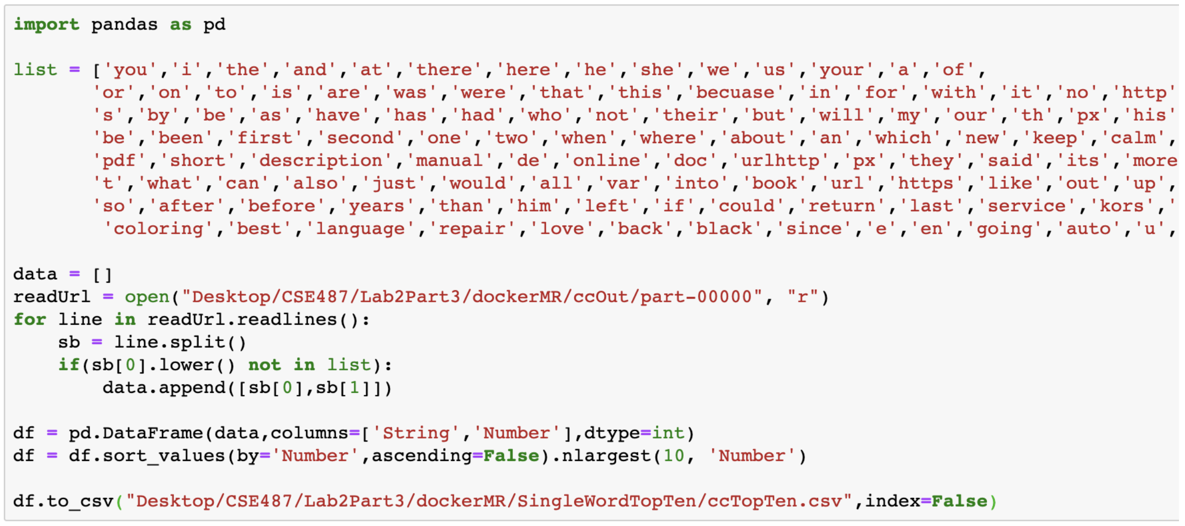
We have done this step during we collect the data (See Step 3 in section 2.1, Step 3 in section 2.2 and Step 4 in section 2.3), But it still have some nature language left. So for here we do a deeper steam and clean.

First, Add some stop word and nature language word into a list.

Second, Read the File which is MR output, and using for loop to iterator each word. If contains a word in the list then remove it.

Third, after end of loop, using pandas library to make a data frame that a data set has 2 variables "word name" and "number", and sort the data set by variable "number" from big to small. Then We take top 10 element, and save as .csv file.

Repeat doing this for Twitter MR Output, New York Times MR output and Common Crwal MR output.



```
import pandas as pd

list = ['you','i','the','and','at','there','here','he','she','we','us','your','a','of',
       'or','on','to','is','are','was','were','that','this','becuase','in','for','with','it','no','http'
       's','by','be','as','have','has','had','who','not','their','but','will','my','our','th','px','his'
       'be','been','first','second','one','two','when','where','about','an','which','new','keep','calm',
       'pdf','short','description','manual','de','online','doc','urlhttp','px','they','said','its','more
       't','what','can','also','just','would','all','var','into','book','url','https','like','out','up',
       'so','after','before','years','than','him','left','if','could','return','last','service','kors','
        'coloring','best','language','repair','love','back','black','since','e','en','going','auto','u',

data = []
readUrl = open("Desktop/CSE487/Lab2Part3/dockerMR/ccOut/part-00000", "r")
for line in readUrl.readlines():
    sb = line.split()
    if(sb[0].lower() not in list):
        data.append([sb[0],sb[1]])

df = pd.DataFrame(data,columns=['String','Number'],dtype=int)
df = df.sort_values(by='Number',ascending=False).nlargest(10, 'Number')

df.to_csv("Desktop/CSE487/Lab2Part3/dockerMR/SingleWordTopTen/ccTopTen.csv",index=False)
```

Figure 13: Deeper Steam and Remove Stop Word and Output a CSV File With Top 10 Word Count

4.2 Visualize Each Output

Step 1 : Download the Tableau from "https://www.tableau.com/" And install into your computer

Step 2 : Open the Tableau Application. In the left side Connect option, Choose Text file and import the word count .csv file we get from section 4.1 .

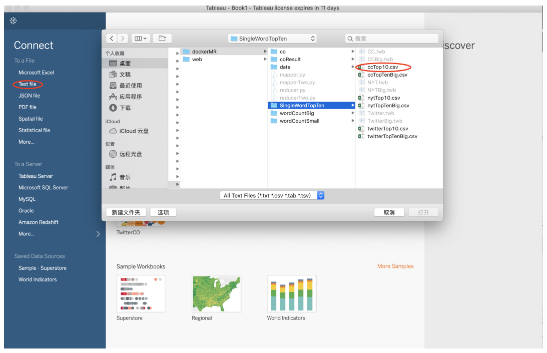


Figure 14: Import Word Count File To Tableau

Step 2 : In the workshop, move String to the text, Number to the Size, and String to the Color. And set Mark show as Text.

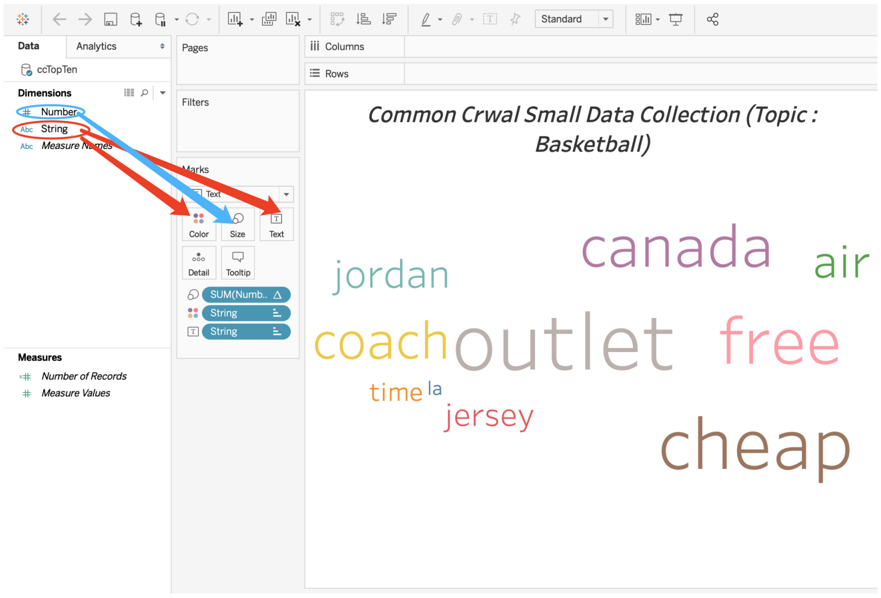


Figure 15: Set up

Step 3 : For the size setting, We want the more appread times word with larger size. So, Change size measure setting as sum, and set discrete.

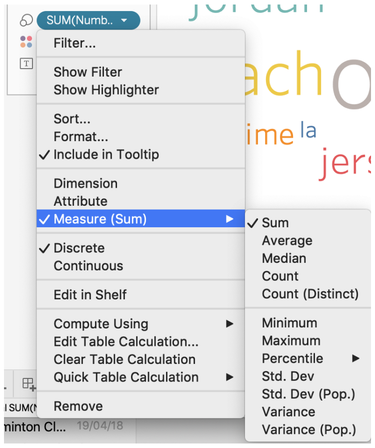


Figure 16: Size Setting

Step 4 : Save the Tableau File, and save the images.

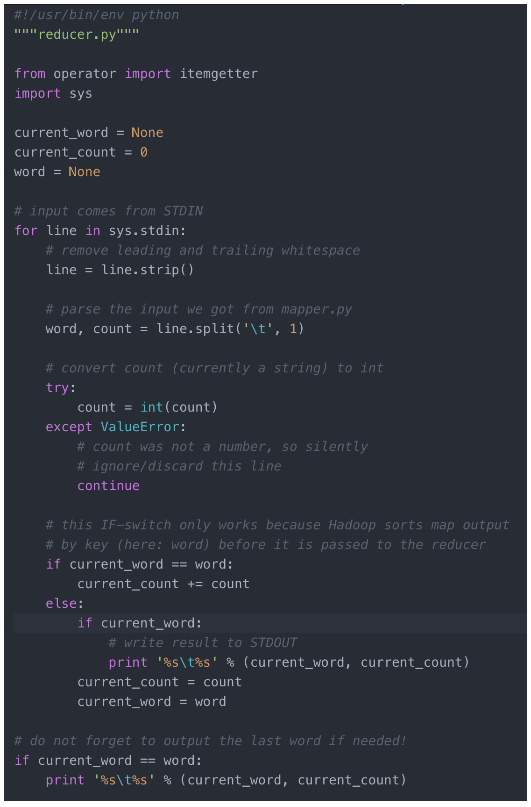
Step 5 : Repeat Step 2 - 4 for all the word count .csv files. And Save those Tableau File and iamges to your local memory. Those Images will be used for later part web design.

4.3 Repeat 4.1 - 4.2 For Big Data Set

Get Big Data : Using Part 1 instruction to collect three different data for a week. And using Part 1 data to run MR which is the instruction of Part 2. And Run 4.1 - 4.2 by MR output.

4.4 Co-Occurrence

Step 1 : Re-write the Mapper and Reducer program. Adding a loop when std read a line for a input. Separate the line into each word. Using two for-loop that find any possible combination with two words, and add 1 into count. Reducer program stay the same.

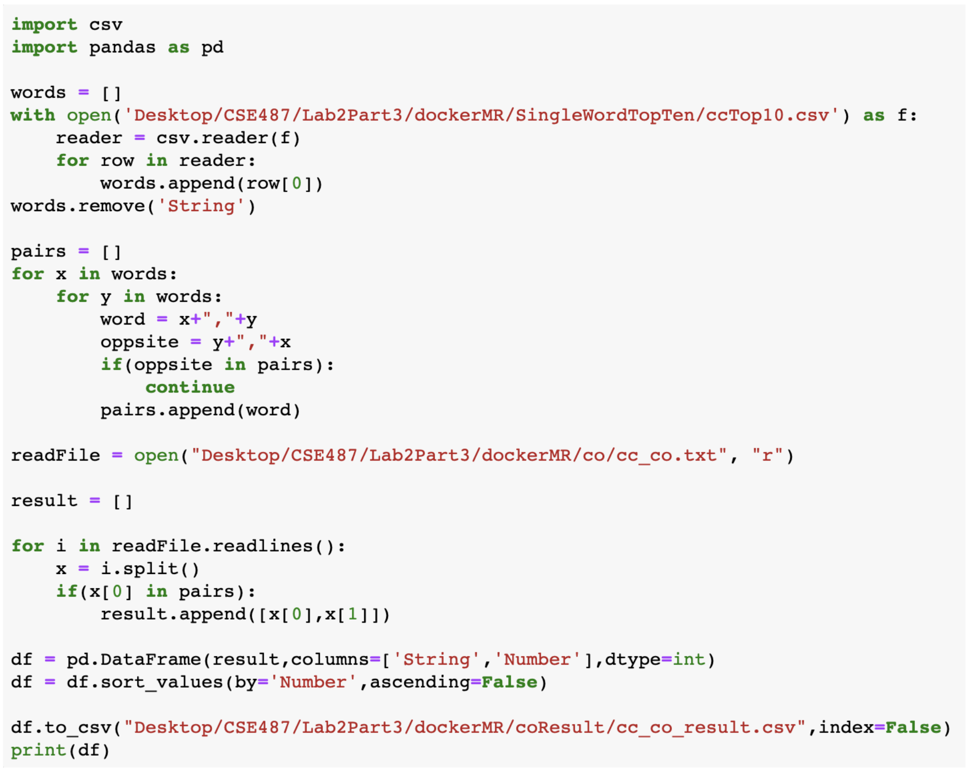


```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Figure 17: Re-write Mapper program

Step 2 : Using instruction Part 2 to run the MR in the cloudera-quickstart with re-write Mapper program. In the output, We get all the possible combination with their count.

Step 3 : Read the .csv File with top 10 words and read the Co-Occureence MR output file, To find all the combinations of ten words and their counts. And Save into CSV. File



```python
import csv
import pandas as pd

words = []
with open('Desktop/CSE487/Lab2Part3/dockerMR/SingleWordTopTen/ccTop10.csv') as f:
    reader = csv.reader(f)
    for row in reader:
        words.append(row[0])
words.remove('String')

pairs = []
for x in words:
    for y in words:
        word = x+","+y
        oppsite = y+","+x
        if(oppsite in pairs):
            continue
        pairs.append(word)

readFile = open("Desktop/CSE487/Lab2Part3/dockerMR/co/cc_co.txt", "r")

result = []

for i in readFile.readlines():
    x = i.split()
    if(x[0] in pairs):
        result.append([x[0],x[1]])

df = pd.DataFrame(result,columns=['String','Number'],dtype=int)
df = df.sort_values(by='Number',ascending=False)

df.to_csv("Desktop/CSE487/Lab2Part3/dockerMR/coResult/cc_co_result.csv",index=False)
print(df)
```

Figure 18: Find Top 10 Words C.O.

Step 4 : Using Instruction In 4.2 To Visualize Output

Step 5 : Repeat Step 2 - Step 4 For Three different Small Data.

4.5 Design A Web

Creating a html web, and add buttons with different. Each button connect to different images (we stored when we visualize each output in Tableau). When click button, will show the image its connected. Then done for this Project.

5 Conclusion

In this project, we learned to use different API to collect data in Python environment and R language environment. Also know a lot of useful and simple environment libraries that help us to finish the project. we also learned the Docker use, Tableau use and simple website design. Data collection is a long term and need a lot of time on it. But after the data collection, the data mining is a very good experience for us to know how to process and analysis the data. And Make a image to show people, behind the data, what the real it is looks like.