# CS5540: Principles to Big Data - SP2017
## Project 1 - Abdullah, Brendan, Noah and Sami
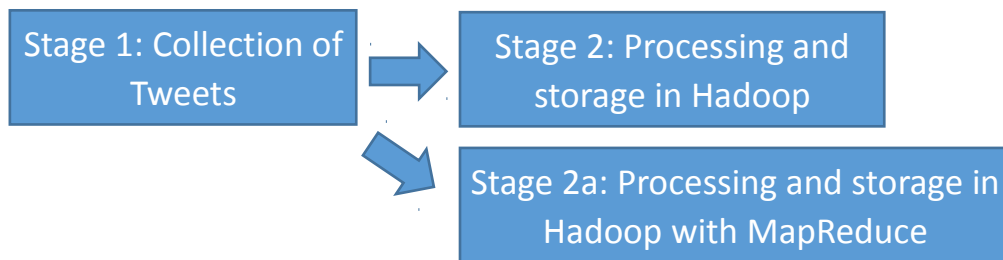
## Project Overview:

The objective of the project is to collect 100,000 tweets from Twitter and identify the top 10 hashtags using the Hadoop file system. This report is split into 5 sections, as follows:

1. Collection of tweets from twitter with Twitter API
2. Processing of tweets to categorize by hashtags and identify top 10
3. Screenshots of Hadoop File System
4. Count Keywords  (Extra Requirement)
5. Appendix: References, Source Code, Etc

The sections will include text, diagrams and source code where applicable to illustrate the methods and results of this project.

From a high level perspective, the algorithm flows as follows:

Stage 1: Collection of Tweets → Stage 2: Processing and storage in Hadoop

Stage 2a: Processing and storage in Hadoop with MapReduce

# Section 1.

As dictated by the project requirements, 100K (100,000) tweets needed to be collected. To accomplish this Stage 1, this project uses the twitter API. By creating a twitter account and setting up an application, tweets can be collected via the Search API. For example, take this query:

https://api.twitter.com/1.1/search/tweets.json?q=the&count=100

This example would return 100 tweets containing the word "the". The tweets will be returned in a JSON format and will need to be processed into t for stage 2.

From a high level perspective, the flow diagram looks like so:
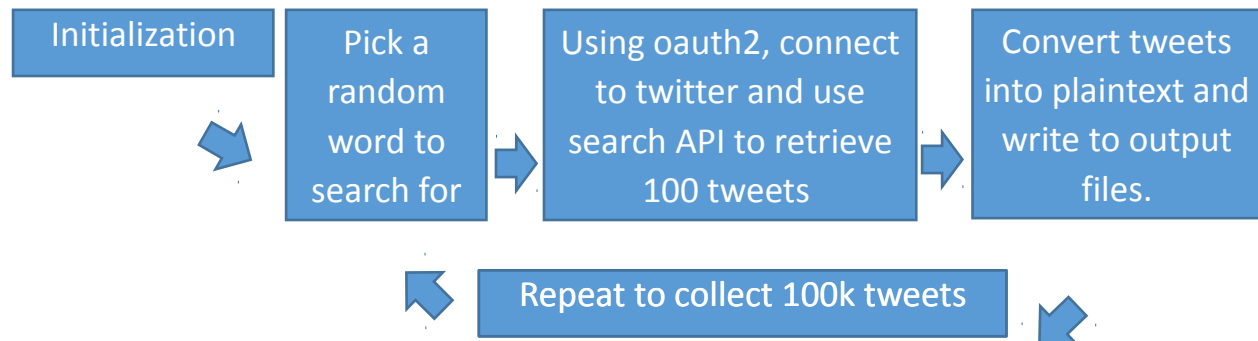


During initialization several libraries are imported, one of which is oauth2. In order to use certain twitter API queries, authentication must first be established. For this project, oauth2 was used; a function, oauth_req, is query twitter. The function will be explained later on.

```python
def oauth_req(url, key, secret, http_method="GET", http_headers=None):
    consumer = oauth2.Consumer(key=CONSUMER_KEY, secret=CONSUMER_SECRET)
    token = oauth2.Token(key=key, secret=secret)
    client = oauth2.Client(consumer, token)
    resp, content = client.request( url, headers=http_headers)
    return content
```

Output files are created and prepared to be written to and a dictionary is opened and stored in a list to facilitate the picking or a random word. In addition, the consumer key and consumer secret are set.

A while loop is started, based on increment variable i. i will increment each time a tweet is written to an output file. A random word is chosen from the list of words that were initialized earlier.

```
# Loop to collect 100k tweets if possible (typically rate limited)
while i < 100000:
    randword = random.choice(word) # random word used to search twitter for tweets
    print str(i)
    print randword
```

With the random word chosen, the program will now query Twitter.

```
home_timeline = oauth_req('https://api.twitter.com/1.1/search/tweets.json?q='+ randword +'&count=100',
    ██████████████████████████████████████████,
    ███████████████████████████ )
tweets = json.loads(home_timeline)  # load tweets into json parser
```

The function will return the tweets to home_timeline. Three arguments are passed into the oauth_req function.

The first is a string concatenated from three parts:

'https://api.twitter.com/1.1/search/tweets.json?q='+ randword +'&count=100'

If for example randword was chosen to be "bye", the concatenated string would be:

'https://api.twitter.com/1.1/search/tweets.json?q=bye&count=100'

The twitter API will understand this to be a request for 100 tweets containing the text "bye". The other two arguments, blacked out, are the token key and token secret. This program is meant to work with a Twitter App, and requires the key and secret for authentication. The earlier defined consumer secret and consumer token are also used for authentication.

home_timeline is then loaded into a JSON parser and stored in tweets. The tweets are converted to string type and then written to output files, one tweet per line. This process, random word, query and write, are repeated until 100,000 tweets are collected.

Write code snippet:

```python
tweets = json.loads(home_timeline)  # load tweets into json parser

# Iterate through tweets and output each tweet to a file, 1 tweet per line
# Split output among 4 files to avoid one giant file
for t in tweets["statuses"]:
    tweet = str(t) + "\n"
    if i % 4 == 0:
        out1.write(tweet)
    elif i % 4 == 1:
        out2.write(tweet)
    elif i % 4 == 2:
        out3.write(tweet)
    elif i % 4 == 3:
        out4.write(tweet)
    i += 1
```

## Section 2.

With the tweets collected, they must now be processed and sorted. Based on the top 10 most common hashtags, 10 categories will be created in the HDFS: one for each hashtag. Two additional categories will also be created to store tweets that either have no hashtags or have a less common hashtag.

From a high level perspective, the flow diagram looks like so:
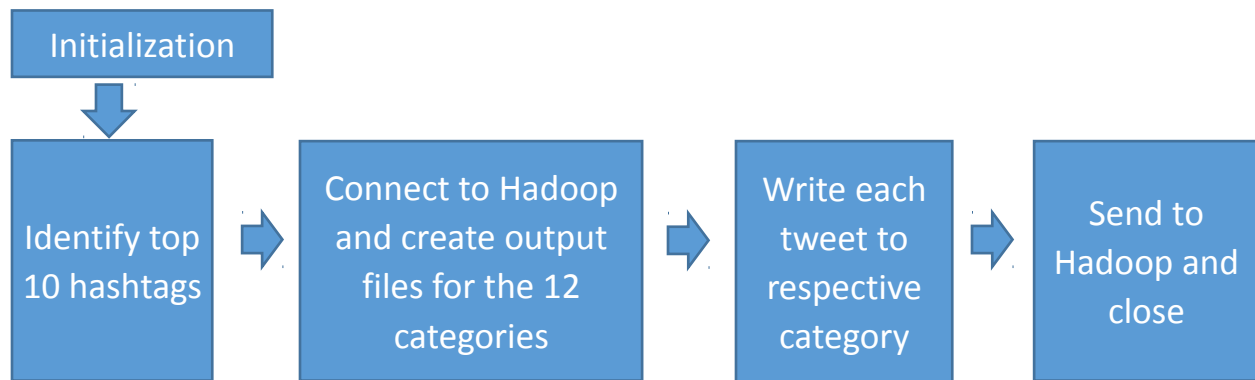


Initialization will import relevant libraries, such as hdfs which is needed to interface with Hadoop.

```python
import ast
import os
import hdfs
import operator


# Helper function to identify which hashtag category a tweet belongs to
def tagCategory(hashtags, word):
    for tup in hashtags:
        tag = tup[0]
        if tag == word:
            return word
    return "OTHER"


# Open tweet input file
fin = open("allout.txt", "r")
tweets = fin.readlines()

hashtagdict = {}
```

The function tagCategory will be explained later. The file containing the tweets is opened and stored in a list. Finally a dictionary is initialized. This dictionary, hashtagdict, will store a count of hashtag occurrences to identify the top 10.

Using a for-loop, each tweet's 'hashtags' property is stored in ht. Since a tweet can have multiple hashtags, a for-loop is used to check each one. If the hashtag is already in the dictionary, simply increment the value. If there is no entry for that hashtag, then set it to one.

After going through every hashtag, in every tweet, the dictionary's items are then sorted so that the entries with the most occurrences are placed at the front. Store those first ten into a list called topten.

```
i = 0
for line in tweets:
    # t = json.loads(ast.literal_eval(line))
    tweet = ast.literal_eval(line)["text"]
    ht = ast.literal_eval(line)["entities"]['hashtags']
    print (str(i) + " : " + tweet)
    if (ht != []):
        h = 1
        for tag in ht:
            print ("hashtag #" + str(h) + " " + tag["text"])
            if (hashtagdict.has_key(tag["text"])):
                hashtagdict[tag["text"]] += 1
            else:
                hashtagdict[tag["text"]] = 1
            h += 1
    i += 1

hashtagsort = sorted(hashtagdict.items(), key=operator.itemgetter(1), reverse=True)
topten = hashtagsort[:10]
```

With a list of the top ten hashtags, the tweets must now be placed into the Hadoop File system. This is accomplished using the hdfs library.

First a connection is established to the HDFS server, which is run locally. Using the topten list, ten directories are created in the HDFS system; a map from the tags to the directories are also created. Two additional directories are set to complete the 12 categories. Output files are then set to store the tweets. For categories of "OTHER" and "NONE", the tags are set directly to the text file.

```python
# Create connection to hadoop and create map from tags to their respective directory in hadoop
tagToHadoopDir = {}
client = hdfs.InsecureClient("http://localhost:50070", user="hduser")
for tup in topten:
    path = "/user/hduser/" + tup[0]
    client.makedirs(path)
    tagToHadoopDir[tup[0]] = path
client.makedirs("/user/hduser/Others")
client.makedirs("/user/hduser/None")

# Create a directory for convince to store category files
if not os.path.exists("tweetfiles"):
    os.makedirs("tweetfiles")

# Create output files for each tweet category
tagToOutFile = {}
for tup in topten:
    strg = "tweetfiles/"+tup[0]+"-tweets.txt"
    tagToOutFile[tup[0]] = open(strg,"w")
tagToOutFile["OTHER"] = open("tweetfiles/Other-tweets.txt","w")
tagToOutFile["NONE"] = open("tweetfiles/None-tweets.txt", "w")
```

The program now begins writing tweets to the appropriate categories. Similar to before, the tweet's hashtags are stored in ht. If ht is null (no hashtag) then write to the file tagged as NONE. This is accomplished using the tagToOutFile function. Otherwise, check each hashtag and write to the appropriate file.

```python
# Write each tweet to their respective tweet category
# options are one of the top ten, other, or none
for line in tweets:
    ht = ast.literal_eval(line)["entities"]["hashtags"]
    if ht == []:
        tagToOutFile["NONE"].write(line)
    for tag in ht:
        tagcat = tagCategory(topten,tag["text"])
        tagToOutFile[tagcat].write(line)
```

A key function is the tagCategory function, which was defined earlier. If the tweet's hashtag is among the top ten, it will be returned to be used as a tag. If the

tweet's hashtag is not among the top ten, "OTHER" will be returned to be used as a tag.

```python
# Helper function to identify which hashtag category a tweet belongs to
def tagCategory(hashtags, word):
    for tup in hashtags:
        tag = tup[0]
        if tag == word:
            return word
    return "OTHER"
```

After all tweets have been written to the appropriate files, the program will tell Hadoop to close the files and finish up.

```python
# Send to hadoop and close each file
for tag, file in tagToOutFile.iteritems():
    if tag == "OTHER":
        hdir = "/user/hduser/Others"
    elif tag == "NONE":
        hdir = "/user/hduser/None"
    else:
        hdir = tagToHadoopDir[tag]
    client.upload(hdir, file.name)
    file.close()

print (client.list("/user/hduser/"))
fin.close() # close tweets input file
```

With the program complete, the project's main requirement has been fulfilled.

# Section 3.

Screenshots of the HDFS after main program completion.

Tweets with the hashstag "Hiring"

× – □   hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1Tr

  hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1TrendingHTs 81x46

102.041524, 36.992427], [-102.041524, 41.003445], [-109.060257, 41.003445]]]}, u'
contained_within': [], u'country_code': u'US', u'attributes': {}, u'id': u'e21c8e
4914eef2b3', u'name': u'Colorado'}, u'metadata': {u'iso_language_code': u'en', u'
result_type': u'recent'}}
{u'contributors': None, u'truncated': False, u'text': u'See our latest #Madison,
WI #job and click to apply: Janitorial Flexible Schedule Premiere Accounts - http
s://t.co/SZybLhvYZk #Hiring', u'is_quote_status': False, u'in_reply_to_status_id'
: None, u'id': 830508167893118976, u'favorite_count': 0, u'entities': {u'symbols'
: [], u'user_mentions': [], u'hashtags': [{u'indices': [15, 23], u'text': u'Madis
on'}, {u'indices': [28, 32], u'text': u'job'}, {u'indices': [126, 133], u'text':
u'Hiring'}], u'urls': [{u'url': u'https://t.co/SZybLhvYZk', u'indices': [102, 125
], u'expanded_url': u'http://bit.ly/2kbKX8e', u'display_url': u'bit.ly/2kbKX8e'}]
}, u'retweeted': False, u'coordinates': {u'type': u'Point', u'coordinates': [-89.
2894975, 43.1581667]}, u'source': u'<a href="http://www.tweetmyjobs.com" rel="nof
ollow">TweetMyJOBS</a>', u'in_reply_to_screen_name': None, u'in_reply_to_user_id'
: None, u'retweet_count': 0, u'id_str': u'830508167893118976', u'favorited': Fals
e, u'user': {u'follow_request_sent': False, u'has_extended_profile': False, u'pro
file_use_background_image': True, u'default_profile_image': False, u'id': 1743742
27, u'profile_background_image_url_https': u'https://pbs.twimg.com/profile_backgr
ound_images/315347754/Twitter-BG_2_bg-image.jpg', u'verified': False, u'translato
r_type': u'none', u'profile_text_color': u'000000', u'profile_image_url_https': u
'https://pbs.twimg.com/profile_images/712227575087570944/SgGJgdNE_normal.jpg', u'
profile_sidebar_fill_color': u'407DB0', u'entities': {u'url': {u'urls': [{u'url':
u'https://t.co/DByWt45HZj', u'indices': [0, 23], u'expanded_url': u'http://www.c
areerarc.com/job-seeker', u'display_url': u'careerarc.com/job-seeker'}]}, u'descr
iption': {u'urls': []}}, u'followers_count': 179, u'profile_sidebar_border_color'
: u'000000', u'id_str': u'174374227', u'profile_background_color': u'253956', u'l
isted_count': 34, u'is_translation_enabled': False, u'utc_offset': -18000, u'stat
uses_count': 62, u'description': u'Follow this account for geo-targeted Facilitie
s Management job tweets in Wisconsin Non-Metro. Need help? Tweet us at @CareerArc
!', u'friends_count': 128, u'location': u'Wisconsin', u'profile_link_color': u'4A
913C', u'profile_image_url': u'http://pbs.twimg.com/profile_images/71222757508757
0944/SgGJgdNE_normal.jpg', u'following': False, u'geo_enabled': True, u'profile_b
anner_url': u'https://pbs.twimg.com/profile_banners/174374227/1458643263', u'prof
ile_background_image_url': u'http://pbs.twimg.com/profile_background_images/31534
7754/Twitter-BG_2_bg-image.jpg', u'screen_name': u'tmj_WI_facmgmt', u'lang': u'en
', u'profile_background_tile': False, u'favourites_count': 0, u'name': u'WI Facil
ity Mgmt.', u'notifications': False, u'url': u'https://t.co/DByWt45HZj', u'create
d_at': u'Tue Aug 03 19:42:08 +0000 2010', u'contributors_enabled': False, u'time_
zone': u'Quito', u'protected': False, u'default_profile': False, u'is_translator'
: False}, u'geo': {u'type': u'Point', u'coordinates': [43.1581667, -89.2894975]},
u'in_reply_to_user_id_str': None, u'possibly_sensitive': False, u'lang': u'en',
u'created_at': u'Sat Feb 11 20:05:51 +0000 2017', u'in_reply_to_status_id_str': N
one, u'place': {u'full_name': u'Wisconsin, USA', u'url': u'https://api.twitter.co
m/1.1/geo/id/7dc5c6d3bfb10ccc.json', u'country': u'United States', u'place_type':
u'admin', u'bounding_box': {u'type': u'Polygon', u'coordinates': [[[-92.889433,

Tweets with no hashtags:

```
× – ☐   hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1Tr

   hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1TrendingHTs 81x46
, u'statuses_count': 666028, u'description': u"ICE-T and COCO interview the TWIN
 TOWERS on FOX-5 / PIX-11 in that link below. Congrat's to us.", u'friends_count':
  899, u'location': u'Anywhere! ', u'profile_link_color': u'1DA1F2', u'profile_ima
ge_url': u'http://pbs.twimg.com/profile_images/829923904256897024/y1axBjl3_normal
.jpg', u'following': False, u'geo_enabled': True, u'profile_banner_url': u'https:
//pbs.twimg.com/profile_banners/101075559/1453444873', u'profile_background_image
_url': u'http://abs.twimg.com/images/themes/theme1/bg.png', u'screen_name': u'Fil
thyBrotherz', u'lang': u'en', u'profile_background_tile': False, u'favourites_cou
nt': 729, u'name': u'THE TWIN TOWERS.', u'notifications': False, u'url': u'https:
//t.co/nbE7A5OIfU', u'created_at': u'Fri Jan 01 22:43:13 +0000 2010', u'contribut
ors_enabled': False, u'time_zone': u'Quito', u'protected': False, u'default_profi
le': True, u'is_translator': False}, u'geo': None, u'in_reply_to_user_id_str': No
ne, u'lang': u'en', u'created_at': u'Sat Feb 11 20:40:32 +0000 2017', u'in_reply_
to_status_id_str': None, u'place': None, u'metadata': {u'iso_language_code': u'en
', u'result_type': u'recent'}}
{u'contributors': None, u'truncated': False, u'text': u'RT @weblackblack: AFRICAN
 AMERICAN, NATIVE AMERICAN, MEXICAN AND CREOLE https://t.co/VIjZoDK6Jz', u'is_quo
te_status': False, u'in_reply_to_status_id': None, u'id': 830516681919303680, u'f
avorite_count': 0, u'entities': {u'symbols': [], u'user_mentions': [{u'id': 36210
59476, u'indices': [3, 16], u'id_str': u'3621059476', u'screen_name': u'weblackbl
ack', u'name': u'blackasf'}], u'hashtags': [], u'urls': [], u'media': [{u'source_
user_id': 3621059476, u'source_status_id_str': u'784719994294853637', u'expanded_
url': u'https://twitter.com/weblackblack/status/784719994294853637/photo/1', u'di
splay_url': u'pic.twitter.com/VIjZoDK6Jz', u'url': u'https://t.co/VIjZoDK6Jz', u'
media_url_https': u'https://pbs.twimg.com/media/CuPih_kWAAEijja.jpg', u'source_us
er_id_str': u'3621059476', u'source_status_id': 784719994294853637, u'id_str': u'
784719833720029185', u'sizes': {u'large': {u'h': 639, u'resize': u'fit', u'w': 63
9}, u'small': {u'h': 639, u'resize': u'fit', u'w': 639}, u'medium': {u'h': 639, u
'resize': u'fit', u'w': 639}, u'thumb': {u'h': 150, u'resize': u'crop', u'w': 150
}}, u'indices': [72, 95], u'type': u'photo', u'id': 784719833720029185, u'media_u
rl': u'http://pbs.twimg.com/media/CuPih_kWAAEijja.jpg'}]}, u'retweeted': False, u
'coordinates': None, u'source': u'<a href="http://twitter.com/download/iphone" re
l="nofollow">Twitter for iPhone</a>', u'in_reply_to_screen_name': None, u'in_repl
y_to_user_id': None, u'retweet_count': 108, u'id_str': u'830516681919303680', u'f
avorited': False, u'retweeted_status': {u'contributors': None, u'truncated': Fals
e, u'text': u'AFRICAN AMERICAN, NATIVE AMERICAN, MEXICAN AND CREOLE https://t.co/
VIjZoDK6Jz', u'is_quote_status': False, u'in_reply_to_status_id': 784291052282798
080, u'id': 784719994294853637, u'favorite_count': 113, u'entities': {u'symbols':
 [], u'user_mentions': [], u'hashtags': [], u'urls': [], u'media': [{u'expanded_u
rl': u'https://twitter.com/weblackblack/status/784719994294853637/photo/1', u'dis
play_url': u'pic.twitter.com/VIjZoDK6Jz', u'url': u'https://t.co/VIjZoDK6Jz', u'm
edia_url_https': u'https://pbs.twimg.com/media/CuPih_kWAAEijja.jpg', u'id_str': u
'784719833720029185', u'sizes': {u'large': {u'h': 639, u'resize': u'fit', u'w': 6
39}, u'small': {u'h': 639, u'resize': u'fit', u'w': 639}, u'medium': {u'h': 639,
u'resize': u'fit', u'w': 639}, u'thumb': {u'h': 150, u'resize': u'crop', u'w': 15
0}}, u'indices': [54, 77], u'type': u'photo', u'id': 784719833720029185, u'media_
```

Tweets from "OTHER" category:

hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1Tr

hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1TrendingHTs 81x46

u'metadata': {u'iso_language_code': u'en', u'result_type': u'recent'}}
{u'contributors': None, u'truncated': False, u'text': u"Have you heard \u2018D@Z'
S  HOUSE PARTY MIX 2017\u2019 by @RC@NE on #SoundCloud? #np https://t.co/wqsytQKz
Ol", u'is_quote_status': False, u'in_reply_to_status_id': None, u'id': 8305175435
29959428, u'favorite_count': 0, u'entities': {u'symbols': [], u'user_mentions': [
], u'hashtags': [{u'indices': [58, 69], u'text': u'SoundCloud'}, {u'indices': [71
, 74], u'text': u'np'}], u'urls': [{u'url': u'https://t.co/wqsytQKzOl', u'indices
': [75, 98], u'expanded_url': u'https://soundcloud.com/daz-jones-4/dzs-house-part
y-mix-2017?utm_source=soundcloud&utm_campaign=share&utm_medium=twitter', u'displa
y_url': u'soundcloud.com/daz-jones-4/dz\u2026'}]}, u'retweeted': False, u'coordin
ates': None, u'source': u'<a href="http://twitter.com" rel="nofollow">Twitter Web
 Client</a>', u'in_reply_to_screen_name': None, u'in_reply_to_user_id': None, u'r
etweet_count': 0, u'id_str': u'830517543529959428', u'favorited': False, u'user':
 {u'follow_request_sent': False, u'has_extended_profile': False, u'profile_use_ba
ckground_image': False, u'default_profile_image': False, u'id': 2879430897, u'pro
file_background_image_url_https': u'https://abs.twimg.com/images/themes/theme1/bg
.png', u'verified': False, u'translator_type': u'none', u'profile_text_color': u'
000000', u'profile_image_url_https': u'https://pbs.twimg.com/profile_images/78979
6112219643905/Aw0VPJt6_normal.jpg', u'profile_sidebar_fill_color': u'000000', u'e
ntities': {u'description': {u'urls': []}}, u'followers_count': 212, u'profile_sid
ebar_border_color': u'000000', u'id_str': u'2879430897', u'profile_background_col
or': u'000000', u'listed_count': 4, u'is_translation_enabled': False, u'utc_offse
t': 0, u'statuses_count': 1416, u'description': u'Bit of a chancer, an a shit dan
cer, bedroom dj, trainee wizard, lurker and all round fuck-up. ROCK DANCE INDIE \
u270c      Y.N.W.A.', u'friends_count': 284, u'location': u'scouser in wigan', u'
profile_link_color': u'FA743E', u'profile_image_url': u'http://pbs.twimg.com/prof
ile_images/789796112219643905/Aw0VPJt6_normal.jpg', u'following': False, u'geo_en
abled': True, u'profile_banner_url': u'https://pbs.twimg.com/profile_banners/2879
430897/1477137023', u'profile_background_image_url': u'http://abs.twimg.com/image
s/themes/theme1/bg.png', u'screen_name': u'JonzeeD', u'lang': u'en', u'profile_ba
ckground_tile': False, u'favourites_count': 1104, u'name': u'D@Z-J-', u'notificat
ions': False, u'url': None, u'created_at': u'Sun Nov 16 11:58:48 +0000 2014', u'c
ontributors_enabled': False, u'time_zone': u'Dublin', u'protected': False, u'defa
ult_profile': False, u'is_translator': False}, u'geo': None, u'in_reply_to_user_i
d_str': None, u'possibly_sensitive': False, u'lang': u'en', u'created_at': u'Sat
Feb 11 20:43:07 +0000 2017', u'in_reply_to_status_id_str': None, u'place': None,
u'metadata': {u'iso_language_code': u'en', u'result_type': u'recent'}}
{u'contributors': None, u'truncated': False, u'text': u"#NP on air  Spargo - you
and me - listen https://t.co/4DN2fNSIjl  -  #dance80 #entertainment #webradio #Sp
argo #SaveTheVinyl #Hit's #over", u'is_quote_status': False, u'in_reply_to_status
_id': None, u'id': 830516858071695360, u'favorite_count': 0, u'entities': {u'symb
ols': [], u'user_mentions': [], u'hashtags': [{u'indices': [0, 3], u'text': u'NP'
}, {u'indices': [69, 77], u'text': u'dance80'}, {u'indices': [78, 92], u'text': u
'entertainment'}, {u'indices': [93, 102], u'text': u'webradio'}, {u'indices': [10
3, 110], u'text': u'Spargo'}, {u'indices': [111, 124], u'text': u'SaveTheVinyl'},
 {u'indices': [125, 129], u'text': u'Hit'}, {u'indices': [132, 137], u'text': u'o

# Section 4.

An extra requirement of the project is to implement a program that counts the number of times a keyword appears in either the tweet's text or hashtag. For this project, a MapReduce algorithm was used.

```
× – ☐   hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1Tr
    hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1TrendingHTs 81x46
hduser@brendan-Q500A:/home/brendan/PycharmProjects/PBDMProject1TrendingHTs$ hadoo
p jar hadoop-streaming-2.7.3.jar -mapper map.py -reducer reduce.py -input /user/h
duser/* -output Output
```

The MapReduce code is split across two files: one for the mapper and the other for the reducer. The input will pull from the 12 directories created earlier. For this project, the keyword "Amazon" was used.

From a high level perspective, the flow diagram looks like so:
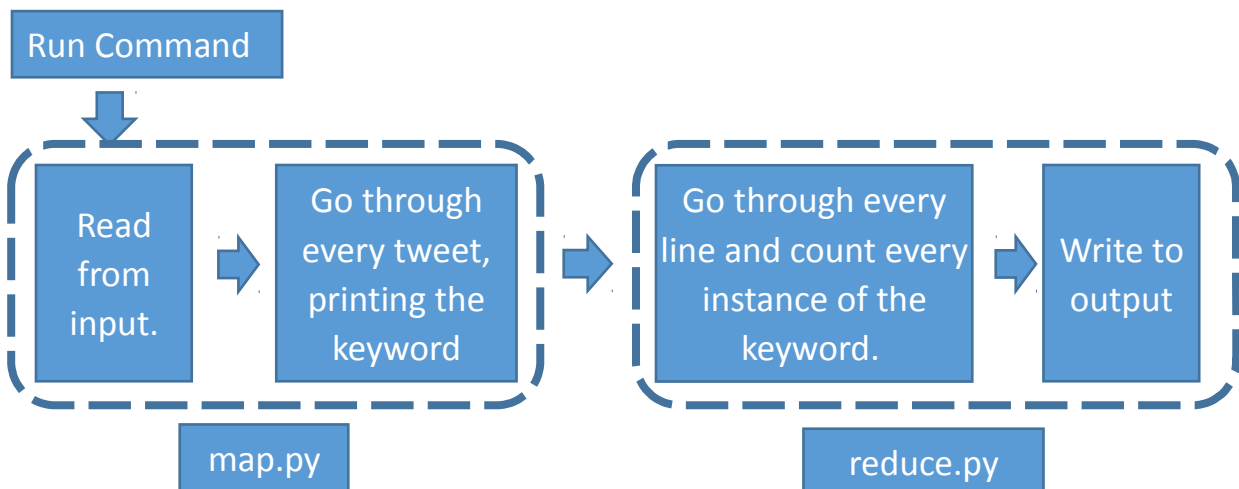


For the mapper, it uses a for-loop is used to read in the tweets from the input directories.

```python
import sys, ast


keyword = "Amazon"
attribute = "hashtags"


for line in sys.stdin:
```

Next, each tweet is checked for the relevant keywords. For this project, attribute is set to "hashtag", so only the hashtag portion will be checked for the keyword. The tweet is parsed down to the hashtag portion, and each hashtag is checked against the keyword. If it matches, it will print.

```python
if attribute == "text":
    try:
        newline = ast.literal_eval(line)["text"]
        newline = newline.strip()
        words = newline.split()
        for word in words:
            if word == keyword:
                print '%s\t%s' % (word, 1)
    except:
        continue


elif attribute == "hashtags":
    try:
        hts = ast.literal_eval(line)["entities"]["hashtags"]
        if not hts == []:
            for ht in hts:
                if ht["text"] == keyword:
                    print '%s\t%s' % (ht["text"], 1)
    except:
        continue
```

For the reducer, it uses a for-loop to read from the prints made by the mapper.

```python
current_word = None
current_count = 0
word = None
```

Three variables are set: current_word and current_count will hold the final count, and will be printed to output.

```python
for line in sys.stdin:
```

For each line, it is processed so that the word ("Amazon") and count ("1") are placed into variables. count is also converted into an int.

```
line = line.strip()

word, count = line.split('\t', 1)

try:
    count = int(count)
except ValueError:
    continue
```

In brief, the counting portion is taken care of by an if/else. Since current_word is initially set to none, the else conditional will trigger. This will set the current_count and current_word to the inputted line. From there, future iterations should trigger the if conditional, updating the current_count. Once all lines have been read in, a final print will be made. The program is complete and has counted every keyword appearance in the tweet's hashtag. (The mapper was set to only check the hashtag)

```
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word


if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Below are two screenshots of the console after the command was executed. At the bottom of the second screenshot, the final result can be seen; it is indicated by a blue arrow.

First Screenshot

```
×  −  ▫   hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1Tr
    hduser@brendan-Q500A: /home/brendan/PycharmProjects/PBDMProject1TrendingHTs 81x46
17/02/15 19:39:02 INFO mapreduce.Job:  map 100% reduce 100%
17/02/15 19:39:02 INFO mapreduce.Job: Job job_local989523038_0001 completed succe
ssfully
17/02/15 19:39:02 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=2262396
                FILE: Number of bytes written=6700024
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=9127596032
                HDFS: Number of bytes written=11
                HDFS: Number of read operations=497
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=18
        Map-Reduce Framework
                Map input records=133291
                Map output records=100
                Map output bytes=900
                Map output materialized bytes=1190
                Input split bytes=1709
                Combine input records=0
                Combine output records=0
                Reduce input groups=1
                Reduce shuffle bytes=1190
                Reduce input records=100
                Reduce output records=1
                Spilled Records=200
                Shuffled Maps =15
                Failed Shuffles=0
                Merged Map outputs=15
                GC time elapsed (ms)=96
                Total committed heap usage (bytes)=7905738752
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=652390400
        File Output Format Counters
                Bytes Written=11
17/02/15 19:39:02 INFO streaming.StreamJob: Output directory: Output
hduser@brendan-Q500A:/home/brendan/PycharmProjects/PBDMProject1TrendingHTs$ ▮
```

Second screenshot – see blue arrow for final result

# Section 5.

**References:**

- Twitter API:
  - https://dev.twitter.com/docs
- Twitter Search API:
  - https://dev.twitter.com/rest/public/search
- Twitter Authentication API:
  - https://dev.twitter.com/oauth/application-only
- OAuth2 Python Library:
  - https://github.com/joestump/python-oauth2
- OAuth example:
  - https://dev.twitter.com/oauth/overview/single-user
- HDFS Install:
  - http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_cluster.php
- HDFS Quickstart:
  - https://hdfscli.readthedocs.io/en/latest/quickstart.html#configuration
- Check if Directory Exists:
  - http://stackoverflow.com/questions/273192/how-to-check-if-a-directory-exists-and-create-it-if-necessary
- Sorting Dictionary:
  - http://stackoverflow.com/questions/613183/sort-a-python-dictionary-by-v`alue
- Hadoop MapReduce
  - https://hadoop.apache.org/docs/r2.6.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/HadoopStreaming.html
- Python/Hadoop MapReduce
  - http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

## Source Code: gettweets.py, gettophts.py, map.py, reduce.py

## gettweets.py:

```
import oauth2
import json
import random

# Twitter provided access key and secret
CONSUMER_KEY = "[REMOVED]"
CONSUMER_SECRET = "[REMOVED]"

# Twitter provided function to submit authenticated requests for data
def oauth_req(url, key, secret, http_method="GET", http_headers=None):
    consumer = oauth2.Consumer(key=CONSUMER_KEY, secret=CONSUMER_SECRET)
    token = oauth2.Token(key=key, secret=secret)
    client = oauth2.Client(consumer, token)
    resp, content = client.request( url, headers=http_headers)
    return content

out1 = open("output1.txt", "a") #output files
out2 = open("output2.txt", "a")
out3 = open("output3.txt", "a")
out4 = open("output4.txt", "a")
words = open("/usr/share/dict/words", "r") #dict of words to random search for
word = words.readlines()
i = 0

# Loop to collect 100k tweets if possible (typically rate limited)
while i < 100000:
    randword = random.choice(word) # random word used to search twitter for tweets
    print str(i)
    print randword

    home_timeline = oauth_req('https://api.twitter.com/1.1/search/tweets.json?q='+ randword
+'&count=100',
                    '[REMOVED]',
                    '[REMOVED]')
    tweets = json.loads(home_timeline)  # load tweets into json parser

    # Iterate through tweets and output each tweet to a file, 1 tweet per line
    # Split output among 4 files to avoid one giant file
    for t in tweets["statuses"]:
        tweet = str(t) + "\n"
        if i % 4 == 0:
```

```python
        out1.write(tweet)
    elif i % 4 == 1:
        out2.write(tweet)
    elif i % 4 == 2:
        out3.write(tweet)
    elif i % 4 == 3:
        out4.write(tweet)
    i += 1

out1.close()
out2.close()
out3.close()
out4.close()
words.close()
```

## gettophts.py:

```python
import ast
import os
import hdfs
import operator


# Helper function to identify which hashtag category a tweet belongs to
def tagCategory(hashtags, word):
    for tup in hashtags:
        tag = tup[0]
        if tag == word:
            return word
    return "OTHER"

# Open tweet input file
fin = open("allout.txt", "r")
tweets = fin.readlines()

hashtagdict = {}

i = 0
for line in tweets:
    # t = json.loads(ast.literal_eval(line))
    tweet = ast.literal_eval(line)["text"]
    ht = ast.literal_eval(line)["entities"]['hashtags']
    print (str(i) + " : " + tweet)
    if (ht != []):
        h = 1
```

```python
    for tag in ht:
        print ("hashtag #" + str(h) + " " + tag["text"])
        if (hashtagdict.has_key(tag["text"])):
            hashtagdict[tag["text"]] += 1
        else:
            hashtagdict[tag["text"]] = 1
        h += 1
  i += 1

hashtagsort = sorted(hashtagdict.items(), key=operator.itemgetter(1), reverse=True)
topten = hashtagsort[:10]

# topten = [(u'nonsenseengine', 530), (u'nonsense', 530), (u'KCAPinoyStar', 280),
#       (u'NadineLustre', 219), (u'job', 217), (u'art', 154), (u'NowPlaying', 136),
#       (u'Hiring', 129), (u'sanremo2017', 126), (u'nowplaying', 120)]

# Create connection to hadoop and create map from tags to their respective directory in hadoop
tagToHadoopDir = {}
client = hdfs.InsecureClient("http://localhost:50070", user="hduser")
for tup in topten:
    path = "/user/hduser/" + tup[0]
    client.makedirs(path)
    tagToHadoopDir[tup[0]] = path
client.makedirs("/user/hduser/Others")
client.makedirs("/user/hduser/None")

# Create a directory for convince to store category files
if not os.path.exists("tweetfiles"):
    os.makedirs("tweetfiles")

# Create output files for each tweet category
tagToOutFile = {}
for tup in topten:
    strg = "tweetfiles/"+tup[0]+"-tweets.txt"
    tagToOutFile[tup[0]] = open(strg,"w")
tagToOutFile["OTHER"] = open("tweetfiles/Other-tweets.txt","w")
tagToOutFile["NONE"] = open("tweetfiles/None-tweets.txt", "w")

# Write each tweet to their respective tweet category
# options are one of the top ten, other, or none
for line in tweets:
    ht = ast.literal_eval(line)["entities"]["hashtags"]
    if ht == []:
        tagToOutFile["NONE"].write(line)
```

```python
    for tag in ht:
        tagcat = tagCategory(topten,tag["text"])
        tagToOutFile[tagcat].write(line)

# Send to hadoop and close each file
for tag, file in tagToOutFile.iteritems():
    if tag == "OTHER":
        hdir = "/user/hduser/Others"
    elif tag == "NONE":
        hdir = "/user/hduser/None"
    else:
        hdir = tagToHadoopDir[tag]
    client.upload(hdir, file.name)
    file.close()

print (client.list("/user/hduser/"))
fin.close() # close tweets input file
```

## map.py:

```python
#!/usr/bin/python
import sys, ast

keyword = "Amazon"
attribute = "hashtags"

for line in sys.stdin:
    if line == "":
        continue

    if attribute == "text":
        try:
            newline = ast.literal_eval(line)["text"]
            newline = newline.strip()
            words = newline.split()
            for word in words:
                if word == keyword:
                    print '%s\t%s' % (word, 1)
        except:
            continue


    elif attribute == "hashtags":
        try:
```

```python
        hts = ast.literal_eval(line)["entities"]["hashtags"]
        if not hts == []:
            for ht in hts:
                if ht["text"] == keyword:
                    print '%s\t%s' % (ht["text"], 1)
    except:
        continue
```

## reduce.py:

```python
#!/usr/bin/python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()

    word, count = line.split('\t', 1)

    try:
        count = int(count)
    except ValueError:
        continue

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```