INF 553 - Spring 2016 Assignment 1

Twitter Sentiment Analysis Using MapReduce

Overview of the Assignment

- Access the Twitter Application Programming Interface(API) using python.
- Estimate the public's perception (the *sentiment*) of the tweet's.
- Calculate the TF-DF(term frequency-document frequency) of terms in a tweet.

Points to keep in mind:

- This assignment is open-ended in several ways. You'll need to make some decisions about how best to solve the problem and implement them carefully.
- **It is perfectly acceptable to discuss your solution on the forum, but don't share code.**
- **Each student must submit his/her own solution to the problem.** The code will be submitted to plagiarism detection software (Moss) to detect if students are submitting the same code.
- You will submit your code, and the professor or grader will run your code against a test data set to see if your code produces the correct answer.
- Your code should only use the Python standard libraries unless you are specifically instructed otherwise. Your code should also not rely on any external libraries or web services.

What you will turn in:

- A file called **lastname_firstname_assignment1.zip** that contains:
  - A text file containing first 20 lines of downloaded twitter data:
    - **lastname_firstname_first20.txt**
  - Program to derive sentiment of each English tweet in downloaded twitter data:
    - **lastname_firstname_tweets_sentiment.py**
  - Program to calculate tf-idf of term occurring in the downloaded twitter data:
    - **lastname_firstname_tweets_tfdf.py**
  - The output from running your tweet sentiment program on your lastname_firstname_first20.txt file:
    - **lastname_firstname_tweets_sentiment_first20.txt**
  - The output from running your tf-idf program on your lastname_firstname_first20.txt file:
    - **lastname_firstname_tweets_tfdf_first20.txt**

The Twitter Application Programming Interface

Twitter provides a very rich REST API for querying the system, accessing data, and control your account. You can read more about the Twitter API (https://dev.twitter.com/docs)

Python environment

If you are new to Python, you may find it valuable to work through the codeacademy Python tutorials. Focus on tutorials 1-9, plus tutorial 12 on File IO. In addition, many students have recommended Google's Python class.

You will need to establish a Python programming environment to complete this assignment. You can install Python yourself by downloading it from the Python website.

Unicode strings

Strings in the twitter data prefixed with the letter "u" are unicode strings. For example:

u"This is a string”
Unicode is a standard for representing a much larger variety of characters beyond the roman alphabet (greek, russian, mathematical symbols, logograms from non-phonetic writing systems such as kanji, etc.)

In most circumstances, you will be able to use a unicode object just like a string.

If you encounter an error involving printing unicode, you can use the encode method to properly print the international characters, like this:

```
unicode_string = u"aaaÃ Ã§Ã§Ã§Ã±Ã±Ã±"

encoded_string = unicode_string.encode('utf-8')

print encoded_string
```

Get Twitter Data

- You will be provided with **twitterstream.py** file which will help you to get live streaming twitter data. You just need to follow the instructions given below after putting in the required Keys.

The steps below will help you set up your twitter account to be able to access the live 1% stream.

- To access the live stream, you will need to install the **oauth2 library** so you can properly authenticate. You can install it yourself in your Python environment. (The command $ pip install oauth2 should work for most environments, or try easy-install oauth2)
- Create a twitter account if you do not already have one. Respond to the resulting email from Twitter to Confirm your Twitter Account.
- Go to **https://dev.twitter.com/apps** and log in with your twitter credentials.
- Click "**Create New App**”
- Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.
- On the next page, click the “**Keys & Access Tokens**” tab along the top, then scroll all the way down until you see the section "Your Access Token”
- Click the button "**Create My Access Token**". You can **Read more about Oauth authorization.** (https://pypi.python.org/pypi/oauth2/)
- You will now copy four values into **twitterstream.py**.
    - "**API Key(Consumer Key)**“
    - "**API secret(Consumer Secret)**“
    - "**Access token**”
    - "**Access token secret**”.
- Open twitterstream.py and you will see the code like below. Replace the corresponding values in the code.

```
api_key = "<Enter api key>"
api_secret = "<Enter api secret>"
access_token_key = "<Enter your access token key here>"
access_token_secret = "<Enter your access token secret here>”
```

- Run the following and make sure you see data flowing and that no errors occur.

    python twitterstream.py > output.txt
    This command pipes the output to a file. Stop the program with Ctrl-C after running it for **1 minute** for data to accumulate. Keep the file output.txt for the duration of the assignment. Don't use someone else's file; we will check for uniqueness in other parts of the assignment.

What to turn in:

lastname_firstname_first20.txt containing the first 20 lines of the twitter data you downloaded from the web. Use the following command for the same.

head -n 20 output.txt > lastname_firstname_first20.txt

Python MapReduce Framework

You will be provided with a python library called **MapReduce.py** that implements the MapReduce programming model. The framework faithfully implements the MapReduce programming model, but it executes entirely on a single machine -- it does not involve parallel computation.

Here is the word count example as a MapReduce program using the provided framework:

```
# Part 1
mr = MapReduce.MapReduce()

# Part 2
def mapper(record):
    # key: document identifier
    # value: document contents
    key = record[0]
    value = record[1]
    words = value.split()
    for w in words:
      mr.emit_intermediate(w, 1)

# Part 3
def reducer(key, list_of_values):
    # key: word
    # value: list of occurrence counts
    total = 0
    for v in list_of_values:
      total += v
    mr.emit((key, total))

# Part 4
inputdata = open(sys.argv[1])
mr.execute(inputdata, mapper, reducer)
```

In Part 1, we create a MapReduce object that is used to pass data between the map function and the reduce function; you won't need to use this object directly.

In Part 2, the mapper function tokenizes each document and emits a set of key-value pairs. The key is a word formatted as a string and the value is the integer 1 to indicate an occurrence of word.

In Part 3, the reducer function sums up the list of occurrence counts and emits a count for word. Since the mapper function emits the integer 1 for each word, each element in the list_of_values is the integer 1.

The list of occurrence counts is summed and a (word, total) tuple is emitted where word is a string and total is an integer.

In Part 4, the code loads the JSON file and executes the MapReduce program which prints the result to stdout.

Submission Details

- For each problem, you will turn in a python script, that solves the problem using the supplied MapReduce framework.
- When testing, make sure **MapReduce.py** is in the same directory as the solution script.
- Your python submission scripts are required to have a mapper function that accepts at least 1 argument and a reducer function that accepts at least 2 arguments. Your submission is also required to have a global variable named mr which points to a MapReduce object.
- If you solve the problems by simply coding the mapper and reducer functions in **tweets_sentiment.py** and **tweets_tfdf.py** then these conditions will be satisfied automatically.
- What you will turn in these **Two** programs:
    1. Program that computes the Sentiment Score of each tweet present in the tweets file called **tweets_sentiment.py**. The sentiment score of a tweet is equivalent to the sum of the sentiment scores for each term in the tweet.
    2. Program that computes tf (term frequency) and df (document frequency) of distinct words in the tweets, called **tweets_tfdf.py**. See the details below for the explanation of tf and df.

- **Finally, to facilitate our grading of your submission, we require that:**
  1. **Your output follow exactly the format as given in the solution files.**
  2. **Add your first name and last name to the beginning of your python scripts, e.g., john_smith_tweets_sentiment.py.**

Problem 1: Derive Tweet Sentiment Score

- For this part, you will use the **MapReduce.py** as framework and **tweets_sentiment.py** as a base for you script.
- We will run your code using the below given command:

  $ python lastname_firstname_tweets_sentiment.py AFINN-111.txt lastname_firstname_first20.txt

- The above given command has two arguments:
  - **AFINN-111.txt**: AFINN-111.txt contains a list of pre-computed sentiment scores. Each line in the file contains a word or phrase followed by a tab separated sentiment score. See the file AFINN-README.txt for more information.
  - **lastname_firstname_first20.txt**: The tweet data file created earlier.

Hint to use AFINN-111.txt:

- To use the data in the AFINN-111.txt file, you may find it useful to **build a dictionary**. Note that the AFINN-111.txt file format is tab-delimited, meaning that the term and the score are separated by a tab character. A tab character can be identified a "\t".
- The following snippet may be useful:

```
afinnfile = open("AFINN-111.txt")
for line in afinnfile:
  term, score  = line.split("\t") # "\t" means "tab character"
  scores[term] = int(score)  # Convert the score to an integer.
```

Your Task:

- Write the **mapper()** and the **reducer()** functions which calculates the Sentiment score of each tweet present in lastname_firstname_first20.txt.
- Each line of lastname_firstname_first20.txt is a streaming message and is represented as a *JSON object*, which stands for JavaScript Object Notation. It is a simple format for representing nested structures of data. It is straightforward to convert this JSON object into a Python data structure using a library called **json**.
- As you can see in the **tweets_sentiment.py** file, there is a call **mr.execute(tweet_data, mapper, reducer)** which makes use of the map-reduce framework. Looking at the **MapReduce.py,** you will see that it imports the json library and calls **json.loads()** on each tweet line. It then calls the **mapper()** and **reducer()** function for each tweet.
- You have to implement the logic for this **mapper()** and **reducer()** function in your **tweets_sentiment.py** file.

Mapper Input:

- The input is a python **dictionary** containing the tweets data: **each_tweet**.
- Extract the '**text**' from this tweet which is now stored in a dictionary. You can also read the Twitter documentation (https://dev.twitter.com/docs/platform-objects/tweets) to understand what information each tweet contains and how to access it, but it's not too difficult to deduce the structure by direct inspection.
- Now that the 'text' for each tweet is available, use that to calculate the sentiment score of each tweet. For that you have to lookup each term of the tweet in the AFINN-111.txt dictionary and add up their respective sentiment scores.
- Keep in mind that all the terms in AFINN-111.txt are lower case. You will have to use the **string.lower()** function to convert your tweet to all lower case for this problem.
- To encode the tweet string in utf-8 and remove ASCII punctuation, import the **string** library and use the following code:
  - **newstring = mystring.encode('utf-8').translate(None, string.punctuation)**

- Some tweets contain **URL's** (eg: https:\/\/t.co\/8wl9wrT), **hashtags**(starting with #,@) and prefix like "**RT @ManUtd:",** which should be ignored while calculating the sentiment score. You can make use of python **regular expression library(import re)** for the same. Also the extra terms found in the tweets but not in AFINN-111.txt and should be given a **sentiment score of 0**.

Reducer Output:

- Your script should **print to stdout** the sentiment of each tweet in the file, one numeric sentiment score per line. The first score should correspond to the first tweet, the second score should correspond to the second tweet, and so on. Please don't sort the scores or the tweets. A sample output for reference is given below:

  [1, 2.0] # Sentiment score of tweet1 is 2

  [2, 0] # Sentiment score of tweet2 is 0

- You can test your solution to this problem using the data file **shortTwitter.txt**:

  python lastname_firstname_tweets_sentiment.py AFINN-111.txt shortTwitter.txt

- You can verify your solution against **tweets_sentiment_shortTwitter_solution.txt**.

What to turn in:

- **lastname_firstname_tweets_sentiment.py** script to calculate the sentiment of tweet.
- **lastname_firstname_tweets_sentiment_first20.txt** containing the output obtained by running your code on your lastname_firstname_first20.txt file.

Problem 2:TF-DF of each term(term frequency – document frequency)

- **Background:** Given a document, ***tf*** (term frequency) of a term in the document is the number of occurrences of the term in the document. For example, consider the following short document D:

  In a swanky hotel ballroom, he told the crowd of alumni and donors that Texas had become the largest feeder of ***students*** *to USC after California, and that* ***students*** *from their state scored significantly higher on the SAT than the average of all applicants.*

- The term "students" occurs twice in the documents, so its tf is 2 for this document.
- Moreover, given a collection of documents, ***df*** (document frequency)of a term refers to the number of documents in the collection where the term occurs. For example, suppose that there are 10 documents in the collection and that "students" occur in three of them, e.g., one of which may be the document D above. Then the document frequency of "students" is 3.
- **tf and df** of a term indicate the importance of the term in the document and across the documents. They are important statistics for effective information retrieval. You may refer to Section 1.3.1 for more details. (Note that df can be used to compute inverse document frequency or IDF)

Your Task:

- For this part, you will use the **MapReduce.py** as framework and **tweets_tfdf.py** as a base for you script.
- We will run your code using the below given command:

  $ python lastname_firstname_tweets_tfdf.py lastname_firstname_first20.txt

- The above given command has one argument:
  - **lastname_firstname_first20.txt**: The tweet data file created earlier.

- You have to implement the logic for this **mapper()** and **reducer()** function in your **tweets_tfdf.py** file.

Mapper Input:

- The input is a python **dictionary** containing the tweets data: **each_tweet**.
- Extract the '**text**' from this tweet which is now stored in a dictionary. You can also read the Twitter documentation (https://dev.twitter.com/docs/platform-objects/tweets) to understand what information each tweet contains and how to access it, but it's not too difficult to deduce the structure by direct inspection.

- Now that the 'text' for each tweet is available, use that to calculate the term frequency(tf) of each term of each tweet.

- Convert the string to lower case before calculating the tf. Also encode the string in utf-8 and remove ASCII punctuation by using:
    - **newstring = mystring.encode('utf-8').translate(None, string.punctuation)**

- Some tweets contain **URL's** (eg: https:\/\/t.co\/8wl9wrT), **hashtags**(starting with #,@) and prefix like "**RT @ManUtd:", "retweet"** which should be ignored while computing the tf-df. You can make use of python **regular expression library(import re)** for the same.

Reducer Output:

The output should be a **list of <term, df, a list of <tweet_no,tf>>** for each term in one line. Note that the framework turns your output tuples into JSON arrays. For example,

["a", 4, [[1, 1], [2, 3], [3, 1], [4, 2]]] # This shows that term "a" has **df** 4 and occurs in tweets 1, 2, 3, 4 with **tf** 1, 3, 1, 2 respectively.

["all", 1, [[3, 1]]] # This shows that term "all" has **df** 1 and occurs in tweet 3 with **tf** 1.

- You can test your solution to this problem using the data file **shortTwitter.txt**:

  python lastname_firstname_tweets_tfdf.py shortTwitter.txt

You can verify your solution against **tweets_tfdf_shortTwitter_solution.txt**.

What to turn in:

- **lastname_firstname_tweets_tfdf.py** script to calculate the tf-df of terms in tweets .
- **lastname_firstname_tweets_tfdf_first20.txt** containing the output obtained by running your code on your lastname_firstname_first20.txt file.