Assignment 3 Twitter Sentiment Analysis using Heuristics

Brief

- **Due date for part a**: 11:59PM 03/13/2016
- **Due date for part b**: 11:59PM 03/28/2016
- **Data**: Stencil (fetch_tweets.py, frequency.py, tweet_sentiment.py, happiest_actor.py, happiest_state.py), Data (user_names.txt, stopwords.txt, AFINN-111.txt, AFINN-README.txt)
- **Handin**: follow the handin instruction
- **Required files for part a**: README_[your NetID], streaming_output_full_[your NetID].txt, streaming_output_short_[your NetID].txt, search_output_[your NetID].txt, fetch_tweets_[your NetID].py, breaking_bad_tweets_[your NetID].csv, frequency_[your NetID].py, reporta_[your NetID]
- **Required files for part b**: README_[your NetID], tweet_sentiment_[your NetID], happiest_actor_[your NetID].py, reportb_[your NetID].txt

---

Part a

Setup

- oauth2. If you machine does not contain such package, run the following command to get the latest package:
    - `sudo pip3 install -U oauth2`
- Note that you are not allowed to use any outside libraries for the whole hw3 except the ones which have been provided in the given python scripts.

To access the Twitter API, you will need to setup a Twitter Developer account.

- Create a twitter account if you do not already have one.
- Go to https://dev.twitter.com/apps and log in with your twitter credentials.

- Click "Create new application"
- Fill out the form and agree to the terms. Put in a dummy website (http://localhost.com, for example) if you don't have one you want to use.
- On tab "Keys and Access Tokens", Click "Create my access token". You can Read more about Oauth authorization.
- Open **fetch_tweets.py** and set the variables corresponding to the consumer key, consumer secret, access token, and access secret.

```
access_token_key = "<Enter your access token key here>"
access_token_secret = "<Enter your access token secret here>"

consumer_key = "<Enter consumer key>"
consumer_secret = "<Enter consumer secret>"
```

Twitter API

Streaming API

To access the current Twitter stream, you need to send a GET request to `https://stream.twitter.com/1.1/statuses/sample.json`. In the stencil code, we already did this for you. If you want to learn more about this API request, go to https://dev.twitter.com/docs/api/1/get/statuses/sample

Go to the directory containing **fetch_tweets.py**, run the following command and make sure you see data flowing and that no errors occur. The program will fetch a small random sample of all public statuses and print out each tweet to the console in a json format. Stop the program with Ctrl-C once you are satisfied.

```
$ python3 fetch_tweets.py -c fetch_samples
```

You can pipe the output to a file, wait a few minutes, then terminate the program to generate a sample. Use the following command:

```
$ python3 fetch_tweets.py -c fetch_samples > streaming_output_full.txt
```

Let this script run for about 5 minutes. Keep the file **streaming_output_full.txt** throughout the assignment. You will be using it in later problems.

What to turn in

Handin **streaming_output_full_[your NetID].txt** along with the first 20 lines of it. You can get the first 20 lines by using the following command:

```
$ head -n 20 streaming_output_full.txt > streaming_output_short_[your NetID].txt
```

Search API

Similar to the Streaming API, to get the tweets related to a search term, you can send a GET request to `https://api.twitter.com/1.1/search/tweets.json`. To learn more about this, visit https://dev.twitter.com/docs/api/1.1/get/search/tweets

Run the following to get the tweets related to a term of your choice

```
$ python3 fetch_tweets.py -c fetch_by_terms -term "[your_chosen_term]" > search_output.txt
```

What to turn in

Choose a search term which is appeared in at least 100 tweets. Handin **search_output_[your NetID].txt**. In your report (reporta_[your NetID].txt), include your search term and some example tweets.

User API

You should be familiar with the Twitter API by now. For this part of the assignment, you will be getting the 100 most recent tweets from each Breaking Bad actors. We've provided the list of their user names (i.e. BryanCranston) in **user_names.txt**. Your job is to implement the function `fetch_by_user_names()` in `fetch_tweets.py`.

To get users' tweets by user name, consult https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline. See the stencil code for **fetch_by_terms** and **fetch_samples** for how to send a request, setup the parameters, and parse the response. You should be able to get the results by running

```
$ python3 fetch_tweets.py -c fetch_by_user_names -file user_names.txt > breaking_bad_tweets.csv
```

Hint

- Even when you set the parameter **count** to 100, the list of tweets returned might contain fewer than 100 tweets because some of them are deleted. Don't worry about this
- To print the text of each tweet, try `print (tweet['text'].encode('utf-8'))`
- To learn more about reading and writing CSV in Python, go here

What to turn in

Handin **fetch_tweets_[your NetID].py** and a csv file **breaking_bad_tweets_[your NetID].csv** with 2 columns: 'user_name', and 'tweet' containing the 100 most recent tweets by the actors listed in **user_names.txt**. For example:

```
user_name,tweet
BryanCranston,"My last tweet was satirical. But Senator McConnell, citizens DO have a voice in selecting Justices. It's called a presidential election."
...
```

Compute term frequency

Write a Python script, **frequency.py**, to compute the term frequency histogram of tweets in **streaming_output_full.txt**

The frequency of a term can be calculate with the following formula:

```
[# of occurrences of the term in all tweets]/[# of occurrences of all terms in all tweets]
```

frequency.py should take a file of tweets as an input and be usable in the following way:

```
$ python3 frequency.py <stopword_file> <tweet_file> > term_freq_[your NetID].txt
```

Assume the tweet file contains data formatted the same way as the livestream data in `streaming_output_full.txt`, and the stop word file contains a list of stop words (for example,

`data/stopwords.txt`). Your program should operate on the text field of tweets, ignoring terms that appear in `stopword_file`.

Your script should output each term-frequency pair, one pair per line, into `term_freq_[your NetID].txt` in the following format:

```
[term] [frequency]
```

For example, if you have the pair (bar, 0.1245) it should appear in the output as (a word and a real number separated by a space):

```
bar 0.1245
```

You should convert all tweets to lower case, and sort your output so that the most frequent terms appear at the top.

Depending on your method of parsing, you may end up with frequencies for hashtags, links, phrases, etc. Some noise is acceptable for the sake of keeping parsing simple. You are free to experiment with strategies for tokenizing, but you will get full credit just by using the default `split()` function (splitting by whitespace characters).

Hint

- Not every tweet dictionary will have a text key -- real data is dirty. Be prepared to debug, and feel free to throw out tweets that your code can't handle to get something working. For example, non-English tweets, or deleted tweets.

What to turn in

1. frequency_[your NetID].py
2. Report the 30 most frequent terms along with their frequencies in reporta_[your NetID].txt

Part b

Determine the sentiment of each tweet

For this part, you will compute the sentiment of each tweet based on the sentiment scores of the terms in the tweet. The sentiment of a tweet is equivalent to the sum of the sentiment scores for each term in the tweet.

You are provided with a skeleton file, tweet_sentiment.py, which can be executed using the following command:

```
$ python3 tweet_sentiment.py <sentiment_file> <tweet_file> > tweet_sentiment_[your NetID].txt
```

The file `data/AFINN-111.txt` contains a list of pre-computed sentiment scores. Each line in the file contains a word or phrase followed by a sentiment score. Each word or phrase found in a tweet, but not in **AFINN-111.txt** should be given a sentiment score of 0. See the file **AFINN-README.txt** for more information.

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("/data/AFINN-111.txt")
scores = {} # initialize an empty dictionary
for line in afinnfile:
    term, score  = line.split("\t")  # The file is tab-delimited. "\t" means "tab character"
    scores[term] = float(score)      # Convert the score to a float.
```

Assume the tweet file contains data formatted the same way as the livestream data.

Your script should output (into `tweet_sentiment_[your NetID].txt`) the pairs of the sentiment and the tweet of the top 10 highest sentiment tweets (in descending order) then the bottom 10 lowest sentiment tweets (in descending order) in the file **streaming_output_full.txt**:

```
[top tweet 1 score]: [top tweet 1 text]
[top tweet 2 score]: [top tweet 2 text]
...
[top tweet 10 score]: [top tweet 10 text]
[bottom tweet 10 score]: [bottom tweet 10 text]
...
[bottom tweet 1 score]: [bottom tweet 1 text]
```

NOTE: The 20 tweets printed out are sorted in descending order on sentiment score. The first sentiment corresponds to the happiest tweet in the input file, the last sentiment corresponds to the unhappiest tweet in the input file.

Feel free to experiment with other methodologies. For example, you can add certain words to the sentiment file such as emoticons (':(', ':)', ...), or acronyms ('lol', 'lmao', ...). You can also try other heuristics that you can find from the user object and the tweet object

What to turn in

1. tweet_sentiment_[your NetID].py
2. Report top 10 and bottom 10 tweets along with their sentiment score in reportb_[your NetID].txt

---

Happiest Breaking Bad actor

Now that you know how to determine the sentiment of each tweet, find out who is the happiest Breaking Bad actor.

Write a Python script, **happiest_actor.py**, that returns the list of user name and average sentiment score pairs sorted in decreasing order on the average sentiment score.

happiest_actor.py should take in a file of tweets in the same format as **breaking_bad_tweets.csv**, a sentiment file (**AFINN-111.txt** for example) as input and be usable in the following way:

```
$ python3 happiest_actor.py <sentiment_file> <csv_file> > happiest_actor_[your NetID].txt
```

Similar to the previous part, your script should output (into `happiest_actor_[your NetID].txt`) the sentiment of all *Breaking Bad actors* sorted in decreasing order on the average sentiment score:

```
[actor 1 score]: [actor 1 username]
[actor 2 score]: [actor 2 username]
...
```

What to turn in

1. happiest_actor_[your NetID].py

2. Report the average sentiment score for each actor in reportb_[your NetID].txt

---

Happiest State

Write a Python script, **happiest_state.py**, that finds the average sentiment score of each U.S. state in the file **streaming_output_full.txt**.

happiest_state.py should take a file of tweets as an input and be usable in the following way:

```
$ python3 happiest_state.py <sentiment_file> <tweet_file> > happiest_state_[your NetID].txt
```

Similar to the previous parts, your script should output (into `happiest_state_[your NetID].txt`) the sentiment of each U.S. state sorted in decreasing order on the average sentiment score:

```
[state 1 score]: [state 1 abbreviation]
[state 2 score]: [state 2 abbreviation]
...
```

Assume the tweet file contains data formatted the same way as the **streaming_output_full.txt**

There are three different objects within the tweet that you can use to determine it's origin.

1. The coordinates object
2. The place object
3. The user object

You are free to develop your own strategy for determining the state that each tweet originates from. (See this for more information about the `tweet` object.) Limit the tweets you analyze to those in the United States.

Hint

- Not every tweet dictionary will have a text key -- real data is dirty. Be prepared to debug, and feel free to throw out tweets that your code can't handle to get something working. For example, non-English tweets, or deleted tweets.

- Your script will not have access to the Internet, so you cannot rely on third party services to resolve geocoded locations.

What to turn in

1. happiest_state_[your NetID].py
2. Report 5 happiest states and 5 unhappiest states along with their sentiment score in reportb_[your NetID].txt

---

Report

For this project, you must write a project report in a text file for each part a and b. In the report you will describe your implementation and any decisions you made to write your algorithm a particular way. Then you will show and discuss the results of your algorithm.

As stated above, we will also be expecting these in your reports:

- For part a (reporta_[your NetID].txt):
    - 30 most frequent terms along with their term frequency
- For part b (reportb_[your NetID].txt):
    - Sentiment scores for for sample tweets (top 10 and bottom 10)
    - Sentiment scores for all Breaking Bad actors
    - The happiest states and the unhappiest states (and their sentiment scores)

Handing in

For submitting to blackboard, create a folder with your NetID, put all the required files in the folder and zip it into **[your NetID].zip**, not other file extension. Please submit the required files for part a to **assignment 3-a**, and the ones for part b to **assignment 3-b** on blackboard.

Late submission is not acceptable. The submission link in Blackboard will disappear immediately at the deadline. You could submit multiple times, only the latest version will be graded.

Cheating/Plagiarism Policy:

Please follow the academic integrity part given as part of the course syllabus.