CS 686 – Data Mining
Lab: Twitter scraping

Due Thursday, Oct 30

Introduction

For this lab, you may work with **one** other person if you wish for the data gathering portion only. At least one of you must be willing to sign up for a Twitter account if you don't already have one, which is required for completing this lab. Details below.

In this lab you will learn how to access Twitter data and try some simple explorations of the results. You will be learning how to use the Twitter API and some of the other new Python libraries we've discussed in class this week. There is a lot more capability available and we will only scratch the surface and hopefully have some fun!

NOTE: You are responsible for following the Twitter developer "Rules of the Road" and following the rate limit guidelines which limit how many queries you may make in a given time period. Before you begin you should at least scan the following: https://dev.twitter.com/overview/terms/policy

I have provided several python files with some initial framework in them in the assignment zip file:

- twitter_access_DM.py – authentication process & some sample API queries.
- twitterstream.py – gets twitter data (for Question 1, below)
- tweet_sentiment.py – compute tweet sentiment (for Question 2, below)

I have also provided a simple sentiment lexicon to use for Question 2, along with a readme file:

- AFINN-111.txt
- AFINN-README.txt

Authentication

The first step is to get authenticated so that you can make queries to the Twitter API. For details or if you run into problems see https://dev.twitter.com/docs/using-search

1. Get a Twitter user account if you don't have one, see: https://twitter.com/
2. Get authenticated by following these instructions: https://dev.twitter.com/docs/auth/tokens-devtwittercom This involves creating an app. Twitter changes this page / approach periodically, but here's what worked for me when I did it this summer:
    - Go to **https://dev.twitter.com/apps** and log in with your twitter credentials.
    - Click "create an application"

- Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use (but has to be a valid URL)
- On the page that appears, go to the API Keys tab, scroll down and click "Create my access token"; it will take a couple minutes (refresh after 1-3 minutes) but then "Your access token" section will show up
- Click the "Test OAuth" button in the upper right-hand corner of the page; you will have to login again to your twitter account.
- Copy your "Consumer key" and your "Consumer secret" into twitter_access_DM.py (the first two missing key/secret pairs)
- Then copy the Access token/token secret into the OAUTH_TOKEN/SECRET variables in twitter_access_DM.py
- Run `twitter_access_DM.py` to test – you should see some trending topics!

Question 1: get some tweets!

First, read this web page to learn more about the different searches you can do on Twitter: https://dev.twitter.com/rest/public/search Play around and decide what search term you are going to use below.

Now you can use twitterstream.py to gather some tweets. You will use these for the remainder of the questions. First, choose a search term that you want to use as the basis of the rest of your analysis. Run the following at the system Terminal to make sure that there are no errors, substituting your chosen search term for `mysearch' below (don't use the quotes!!). Stop the program with Ctrl-C once you are satisfied it's working:

```
$ python twitterstream.py `mysearch'
```

If you are working in pairs, this is the last part to do together. You should each choose a different search term.

Now save the output to a file. The below command pipes the output to a file called output.txt:

```
$ python twitterstream.py `mysearch' > output.txt
```

Let this script run for a minimum of **10 minutes**, using different searches for each of you if working in teams. You should then each take your own file and **Stop working together** for the rest of the questions.

(2 pts) What to turn in: The first 20 lines of your file, called top-20-output.txt. You can get this via the command:

```
$ head –n 20 output.txt > top-20-output.txt
```

Question 2: Derive tweet sentiment

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 once you have filled in the empty method:

```
$ python tweet_sentiment.py <sentiment_file> <tweet_file>
```

The file 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. 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" (check out the `split` method for strings!)

The tweet_file is the output.txt from Question 1 (or one formatted similarly). You will have to do some JSON wrangling here. Check out the Twitter documentation to determine what fields to parse. Your script should print to stdout the sentiment of each tweet in the file, one sentiment per line:

```
<sentiment:float>
```

NOTE: You must provide a score for **every** tweet in the sample file, even if that score is zero.

Note that for this part and all the following questions, you are free to play around with cleaning up the text of the tweets to your heart's content to make the results more robust to the noisy nature of Twitter text! You might want to play around with the tokenizers provided by Chris Potts (see the Readings), for example. (Remember to give attribution for all code downloaded from the web as part of the text processing that you do).

(5 pts) What to turn in: Your python file called `tweet_sentiment.py`

Question 3: Compute term frequency

For this part, you will compute the term frequency histogram of the Tweet data you harvested from Question 1. You may find it useful to consult Example 23 in the Twitter Cookbook from the readings. Name your script `frequency.py` (I have not provided a skeleton).

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 (same format as that from Question 1) and be usable in the following way:

```
$ python frequency.py <tweet_file>
```

Your script should print to stdout each term-frequency pair, one pair per line, in the following format:

```
<term:string> <frequency:float>
```

For example, if you have the pair (bar, 0.1245) it should appear in the output as:

```
bar 0.1245
```

Frequency measurements may take phrases into account, but this is not required. We only ask that you compute frequencies for individual tokens.

Depending on your method of parsing, you may end up with frequencies for hashtags, links, stop words, phrases, etc. Some noise is acceptable for the sake of keeping parsing simple.

(3 pts) What to turn in: Your python file called `frequency.py`

Question 3: Top ten hash tags

Write a Python script, `top_ten.py`, that computes the ten most frequently occurring **hash tags** from the data you gathered in Question 1. (I have not provided a skeleton).

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

```
$ python top_ten.py <tweet_file>
```

In the tweet file, each line is a Tweet object, as **described in the twitter documentation**.
You should not be parsing the "text" field.

Your script should print to stdout each hashtag-count pair, one per line, in the following format:

```
<hashtag:string> <count:float>
```

For example, if you have the pair (baz, 30) it should appear in the output as:

```
baz 30.0
```

Remember your output must contain floats, not ints

(5 pts) What to turn in: Your python file called `top_ten.py`