LINGUIST 492 - HOMEWORK 1

INTRODUCTION

In this assignment you'll be analyzing unfiltered twitter data. You'll learn to do some text processing (tokenization), analyze frequency distributions, and do some basic topic modeling. If you do the extra credit portion, you'll also do some rudimentary language modeling.

The homework assignment comes with three files:

- Instructions.pdf (this document)
- twitterFrequency_stub.py
    - contains the basic structure and some provided code that you will be extending for this assignment
    - make a new copy of this program and call it **twitterFrequency.py**
- a zip file of Corpora, all of which have one tweet per line, created by searching for keywords
    - 'all': 100k tweets using English function words as keywords
    - 'classchoice': 10k tweets with 'movie' keywords we chose in class
    - 'corpus1'-'corpus5': each has 10k tweets created using keywords corresponding to different topics
    - 'mystery' – 1000 tweets using keywords from one of the five topics above

Save these files to a directory on your computer. The provided program should be run on the command line with 1 or more corpus files provided as arguments. For example:

```
>python twitterFrequency.py Corpora/tweets_corpus_all.txt
Corpora/tweets_corpus5.txt Corpora/tweets_corpus_mystery.txt
```

As given, the program will read through each file named on the command line and save a new file with a "`_out.txt`" extension that lists each 'word' in the file along with the number of times it occurs in the file (where 'word' is anything separated by spaces or hyphens surrounded by spaces).

Before you move on, make sure you understand what each piece of code in the program is doing (you don't have to worry about the bits that are marked as 'Extra Credit' for now).

PART 1

In the first part of the assignment, you will write the 'clean()' function that cleans up and tokenizes tweets. This function is called on each line before splitting it into words. Use regular expressions and the `re` module to do the following things (each thing should be one line of code):

- Replace all twitter handles (alphanumeric strings beginning with '@') with the string '@@@'
- Replace all urls with 'www'
- Replace all hashtags (alphanumeric strings beginning with '#') with the string '###'
- Remove all word-external punctuation. This should include things like question marks, commas, quotation marks, ellipses, etc. Don't remove word-internal hyphens (like in "co-worker") or apostrophes (like in "won't"). Use your best judgement to characterize what counts as a "word". It doesn't have to be perfect.
- Remove leading and trailing whitespace on the line.
- Lowercase everything on the line

I suggest using a print statement inside of clean() to compare your lines before and after cleaning so that you can make sure your function is doing what you want. Once your clean() function is working, remove or comment out the print statement. As a sanity check, once you 'clean' your data, the number of word types in corpus4 should go roughly from 28k to 14k.

PART 2

In this part you will implement the normalize() function. This should take about 3 lines of code. The normalize function takes a dictionary of word-frequency pairs, calculates the sum of all the values in the dictionary, and divides each value in the dictionary by that sum. When the script calls this function (in three places), the frequencies in the dictionary that is passed to this function are converted to relative frequencies (e.g. proportions) that sum to 1. Since the get_freqs() function calls normalize(), completing the normalize() function will cause your program to print relative frequencies (expressed as decimals between 0 and 1), rather than counts for each file.

Questions
Run your program on all of the corpora (you can run it just once, specifying all the corpus file names at the command line). Examine the relative frequency distributions for all of the corpora. For each of corpus 1-5 and the 'mystery' corpus, what are some high frequency words that seem to be unique to each corpus? That is, what are words whose frequencies are unusually high in each corpus? Using this information, can you guess for each corpus what topic it corresponds to?

PART 3

In this part of the assignment you will extend the save_histogram() function. As provided, the save_histogram() function prints the (relative) frequency for each word. Modify the save_histogram() function to add two additional fields to each line: the log frequency and the log rank of each word. To make these as readable as possible, print these as floats separated by tabs after the word (e.g. add '\t%f\t%f' to the formatted string. You will need to look up how to compute logs and import the appropriate library. It doesn't matter what base the log is (as long as it's the same base for both), and it doesn't matter if you take the log of the count or the relative frequency.

Questions
Run your program again on the 'all' corpus to find out whether twitter data follow Zipf's law. If twitter data follows Zipf's law, then log frequency and log rank should be linearly related (see Exercise 9 in Chapter 13 of TP for more details). To see if log frequency and log rank are linearly related, open your '…_out.txt' file in Excel or Google Spreadsheets or your favorite plotting software and plot a scatter plot for the log frequency and log rank fields. Does twitter data seem to follow Zipf's law for natural language? Explain. Save your plot as an image and attach it to your homework submission.

PART 4

In the final part of the assignment, you will write the get_top() function, use it to collect stop words from the 'all' corpus, and filter the words from the other corpora to ignore the stop words. Write the get_top() function: it takes a dictionary of word-frequency pairs and an integer N, and it should return a list of the N most frequent words in that dictionary. This should take 5 or 6 lines of code.

Once you have get_top() working, uncomment the 'filter(…' line on line 74 of the code. Once this line of code is active, the program will collect the top N most frequent words from the first file and remove them from the dictionaries of all the other files specified on the command line.

Questions
Choose a value of N by examining the word frequencies in the 'all' corpus. Choose the highest value of N that includes only common function words and no content words relating to the topics of the corpora. Re-run the program on all the files, specifying the 'all' file as the first argument, and all the others after it.

LINGUIST 492 - HOMEWORK 1

Examine the resulting word frequency files for corpora 1-5 and the mystery corpus after removing the stop words and answer the following questions:

a) What value of N works well?
b) After removing stop words, are the topics clearer? Do you have clearer guesses for what keywords I used to generate the corpora? Do these results confirm the topics you guessed in Part 2? Explain.
c) Did we need to use multiple corpora? That is, did we need to collect stop words from the 'all' corpus and extract them from the other corpora? Would this technique have worked if we had just removed the top N words from each corpus? Why or why not?
d) If you had to write a program to automatically determine which of topics 1-5 a mystery file came from, how could you do it? You don't need to write the program, just explain in a few sentences what the program could do.

EXTRA CREDIT

The provided script comes with some code for generating random tweets from each corpus. To see these random tweets (they will print to standard output), simply uncomment lines 77 through 86. These lines of code will generate 5 random tweets for each file provided on the command line after the first file.

For extra credit, explain how the program is generating the random tweets, and comment on the quality of the tweets. How do these random tweets differ from actual tweets (i.e. what information is the model lacking)?

WHAT TO SUBMIT

- Submit (as an attachment) your program '**twitterFrequency.py**' with the extensions from Parts 1-4.
- Submit your answers to questions in Parts 2-4 as an attachment (either in plain text or pdf).
- Submit your plot from Part 3 as an attachment