MTH 3300
Spring 2019
E. Fink
April 30, 2019

Homework 10

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load these template files, and write the following programs. Submit your source code to me via Blackboard, in `.py` format; do NOT send any other files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

Be sure to read the SPECIFICATIONS carefully for all problems! And write comments!

1) airlines.py

I have a spreadsheet of real tweets sent from people to a number of corporate airline accounts. The spreadsheet, which is an edited version of one that is available on Kaggle (`www.kaggle.com`), also contains a number of pieces of information about each tweet, including one called `airline_sentiment`. For this column, a human has laboriously read each tweet, and came to a conclusion about whether the tweet expresses a `positive`, `neutral`, or `negative` sentiment about the airline. For example, `@united go to hell` would be marked as a `negative` sentiment.

I am a brand manager for an airline, and I need to follow the public's tweets. But I can't sit around reading Twitter all day, and so I would like to have a computer identify sentiments for me. But how can the computer decide whether a tweet is positive or negative?

After perusing this data set, I have come up with a (*verrrrry* unscientific) procedure for a program to predict the sentiment of a tweet. The procedure is based on the following formulas:

$$f_{pos} = 0.93n_{great} + 0.87n_{thank} + 0.71n_{thanks} + 0.53n_{love} + 0.46n_{appreciate}$$

$$f_{neut} = 0.74n_{do} + 0.68n_{add} + 0.62n_{rt} + 0.51n_{dm} + 0.39n_{can} + 0.26n_{please}$$

$$f_{neg} = 0.35 + 1.2n_{worst} + 1.08n_{hours} + 0.97n_{delay} + 0.43n_{no} + 0.41n_{why}$$

Here, for example, n_{great} refers to the number of times the word `great` appears in the tweet, n_{thank} is the number of times the word `thank` appears in the tweet, etc. (Of course, most of these n's will be 0 for most tweets!)

Each *tweet* has its own f_{pos}, f_{neut} and f_{neg} value. Given a tweet, our procedure to predict the sentiment of a tweet is to first `.split()` it into "words" (more properly, *tokens*); compute the values of f_{pos}, f_{neut}, f_{neg} for the entire tweet; and then output the sentiment corresponding to the highest of these 3 numbers (e.g., if $f_{pos} = 0.57, f_{neut} = 0.91$ and $f_{neg} = 0.72$, then the program would predict the sentiment `neutral`).

Question: using this procedure on my 4000 tweets, what percent of the tweets does it label accurately? (This would give a good estimate of how well it would guess the sentiment of airline tweets in general, even new ones that no human had sat down to label.)

Your first mission is to create three dictionaries called `pos_coeffs`, `neut_coeffs`, and `neg_coeffs`. The keys in these dictionaries will be the 5 strings in the files `poswords.txt`, `neutwords.txt` and `negwords.txt`, respectively; the values will be the coefficients next to them. You can hard code the dictionary if you like (i.e., type in the values one-by-one from the file directly into your code).

Next, create a function called `predict_sentiment`. This function should receive one string called `tweet`, and it should return one string called `sentiment`. As the names indicate, the argument `tweet` should be a string containing an entire tweet, and the output should be one of the strings `positive`, `neutral`, or `negative`. The function should first break down the `tweet` using `.split()`. Then, it should compute the three sentiment scores f_{pos}, f_{neut} and f_{neg} by applying the three dictionaries to each word in the sentence (for example, every time you see the word `worst`, you add 1.2 to f_{neg}, since the coefficient of n_{worst} is 1.2, and you add 0 to f_{pos} and f_{neut} since there is no term for `worst` in these scores). Finally, the function should compare these three scores, and return the appropriate string as output (ties should be given to `neutral`).

Since f_{pos}, f_{neut} and f_{neg} are based on word counts, we should be clear about how words get counted. n_{great} should count all occurrences of the word `great` with any capitalization: `great`, `Great` and `GREAT` should all count. However, if the word `great` appears with any punctuation (e.g. `great!`), or as part of a longer word (e.g. `greattttttt`), your program is allowed to ignore it. To deal with case differences, the `.lower()` method can help: this automatically turns each uppercase letter in a string to its lowercase version. (So `"Great".lower() == "great".lower()` would be `True`.)

Finally, create a Pandas dataframe from the spreadsheet `tweets.csv`. For each of the 4000 tweets in the dataframe, identify its actual sentiment (contained in the spreadsheet) and its predicted sentiment (according to the function you wrote), and determine the number of tweets where the predicted sentiment agrees with the actual sentiment. This number should be your program's final output.

Specifications: your program must

- create and use three dictionaries, as describe above.
- create a Pandas dataframe from the `tweets.csv` spreadsheet.
- calculate f_{pos}, f_{neut} and f_{neg} for each Tweet in the dataframe, as described above.
- calculate the number of Tweets for which the predicted sentiment from these numbers matches the actual sentiment stored in the spreadsheet.

***Challenge**: come up with a more accurate procedure to predict sentiments! It can use any data in the spreadsheet in its predictions, aside from columns B, C, D and E.*