CS 101
Program 5
Algorithm due : Oct 23rd 2016
Program due : Oct 30th 2016

Who (could have) said that?

One application of computer technology is textual analysis: conducting detailed analysis of various documents to look for patterns that a human would overlook, by looking at word choices or other statistical properties of the text. These techniques have been used to settle questions of authorship or mistaken attribution, to detect forgery, and to analyze how language has changed over time.

For this program you will be given the text of 4 different political speeches. You are also given the text of 4 “unknown” speeches; your task is to report which of the known speeches the unknown texts most closely resemble. You will use 2 different measures of similarity: word commonality and word frequency.

Both of these techniques begin by “cleaning” the text in order to ensure better results. Because you will carry out the same actions on all files, this should be put into a function.

- The text is converted to a single-case, either all upper or all lower. (Lower is easier to read but it doesn’t really matter.)
- All punctuation marks are removed. (The string module has a predefined string of just the punctuation characters, called string.punctuation. This will be helpful.)
- The string is broken into a list of smaller strings (words).
- Common ‘stop words’ such as ‘the’, ‘this’, ‘an’, ‘in’, etc. are removed, as they are so commonly used in all kinds of text that their presence only obscures whatever relationships may be present.

The text is then ready to begin analysis. For this program, we will be looking at single words, and not larger groupings such as pairs, sentences, etc.

Once each text has been cleaned and broken into a list of strings, produce for each text a dictionary in which the keys are individual words and the values are the number of times each word appears. We will use these to compare the various texts.

Word Commonality

Word commonality is simply a measure of how many different words there are that appear in all of a set of documents. For this program we’re working with pairs of documents. Suppose document A has 5000 distinct words, and document B has 4000 distinct words. There are 2500 words that appear in both documents. Thus, there are 5000 + 4000 – 2500 = 6500 distinct words in our 2-document set, and the word commonality is 2500/6500 = 38.46%. Suppose we find another document C containing 5100 distinct words, and that it shares 3400 words with document A. Then there are 5000+5100-3400 = 6700 distinct words in the A/C pair, and their word commonality is 3400/6700 = 50.75%, and we can see that document C is more similar to A than B is.

Note that for this analysis we are only looking at how many distinct words appear, regardless of their frequency. If one of the texts includes the word “banana” 100 times, that’s still just one word for this analysis.
Your program should compare each of the unknown texts to each of the known texts, and report which text the unknown text has the highest word commonality score with.

Relative frequency

For this analysis, after we have identified all the words that appear on both texts, we want to see if they appear at about the same frequency; this helps us measure the extent to which the same words are being used the same way.

We already have the count of how many times each word appears. We will use this to determine what proportion of all words is represented by each individual word. An example will make this clearer.

Suppose our text is: “It was the best of times, it was the worst of times.” We start by converting it to all lower-case and removing punctuation, then breaking it into words: “it”, “was”, “the”, “best”, “of”, “times”, “it”, “was”, “the”, “worst”, “of”, “times”. After removing the stop words, we have: “best”, “times”, “worst”, “times”. There are 4 words in this list. Their relative frequencies are: “best”: 0.25; “times”: 0.5; “worst”: 0.25. We divide the frequency of each word’s occurrence with the total number of all words’ occurrence.

We’ll use the *root-mean-square* method to compare them for similarity. (That’s the square root of the average [mean] squared difference.) To compare two texts for similarity using this measure, we need to carry out the following procedure:

- Find all words that the 2 documents have in common (that is, words that appear in both).
- For each such word:
  - Find the difference in relative frequency between the two documents
  - Square that difference
  - Keep a running total of the sum of the squares
- When the sum of the squares for all words is known:
  - Divide the sum by the number of distinct words the documents have in common
  - Find the square root of that quotient.

Note that for this type of analysis, higher similarity is indicated by a *lower* score. If two documents have exactly the same frequencies (meaning one can be produced by rearranging the words in the other), their score by this measure is 0.

You are given 4 “known” texts:

- Acceptance speech by Donald Trump, 2016
- Acceptance speech by Hillary Clinton, 2016
- Acceptance speech by Mitt Romney, 2012
- Acceptance speech by Barack Obama, 2012

You are also given 4 ‘unknown’ texts. For each, report the known text for which they have the highest similarity by each measure.

Design notes:

- Think about your program carefully! There are a lot of steps in this program. Identify areas were you’re doing the same thing, just with different data. Those are prime candidates to turn into functions. This will help you test each part of your code separately, and to manage the mental complexity involved. For example, you may want functions to do some of the following:
    - Take a string and strip out all punctuation characters, returning the lower-case version.
    - Take a list of strings and return a dict where the keys are strings and the values are the frequency counts.
    - Take a dictionary of words and counts, and return a dictionary of words and frequencies.
    - Take 2 dictionaries of words and counts, and returns the word commonality.
    - Several others… think about it!

Sample run:

```
== RESTART: Q:\2016_Fall\CS101\Solutions\Program05\Solution\src\Program5.py ==
The text mystery1 has the highest word commonality with Romney (16.3212%).
The text mystery1 has the highest frequency similarity with Trump (0.0030).

The text mystery2 has the highest word commonality with Romney (22.5907%).
The text mystery2 has the highest frequency similarity with Clinton (0.0019).

The text mystery3 has the highest word commonality with Romney (19.3272%).
The text mystery3 has the highest frequency similarity with Trump (0.0030).

The text mystery4 has the highest word commonality with Obama (22.9307%).
The text mystery4 has the highest frequency similarity with Clinton (0.0023).

>>>
```