CIS 4930 NLP // HW #6 // Spring 2018

Date Assigned: February 11, 2018
Date Due: February 21, 2018

Submission Format

You will submit a soft copy of your solution using e-Learning (http://elearning.ufl.edu) by the end of the day (23:59 / 11:59 PM) on the assigned date (February 21). Submit one file, **hw6.py**.

Assignment

At the top of every solution file you submit this semester include: your name, section number, the assignment number, and the date due. Implement the Python to Complete these exercises.

Exercises

1. It's 2:05am on a Friday night and you're diligently working on your latest NLP assignment, when suddenly your phone beeps to indicate an incoming text message. Opening it, you find the following: "I an really out me it. I lost ox bbq. Can wot bone un het of?". Guessing that your friend was surely trying to impart upon you some great truth about life and the universe, but was so caught up in the excitement of his epiphany that he couldn't be bothered to check the accuracy of his T9 entries (https://en.wikipedia.org/wiki/T9_(predictive_text)), you desperately try to piece together what he might have been saying. Finding it hopeless to manually go through all the possibilities, you decide to write a python program that will automatically detect unlikely words in a sentence and replace them with more probable words that share the same key sequence.

   You will need a CFD to predict the next word in a message given the previous word. To generate this, you will use the Spoken American English (http://www.linguistics.ucsb.edu/research/santa-barbara-corpus) corpus (sbcorpus zipfile – http://www.linguistics.ucsb.edu/sites/secure.lsit.ucsb.edu.ling.d7/files/sitefiles/research/SBC/SBCorpus.zip). Please put the corpus in a directory called *sbcorpus* in the same path as your code. You will need to implement Python code to:

   - go through the directory structure and pull out all the text from all of the *trn* files.
   - use the split command or regular expressions to remove the names and timestamps.
   - build a corpus from these files, get the bigrams and construct a CFD.

   Once you have the CFD, you can simply iterate through the text message and look at the probability of each word given the previous word; if any word is extremely improbable, you should look at possible alternatives and see if any of them are any more probable. If you find a more probable alternative, replace whatever's there and be sure to use the new word to determine what the next word should be. There is an example in Chapter 3 that uses regular expressions to do part of this. You are likely to find it helpful to build a data structure that maps each letter to its corresponding letter bundle (for example 'a' : '2abc').

You should have a function that takes as input a sentence (you can put spaces before periods and other punctuation if helpful) and provides as output a translated sentence. For the purposes of your submission, call your function with the text in the problem statement above. For this test case, your program should finish efficiently (around a minute or less) and be 100% accurate.