CMPUT 396, Winter 2019, Assignment 5

All assignment submissions must conform to the Assignment Submission Specifications *posted on eClass. Ensure that your submission follows these specifications before submitting your work.*

You will produce a total of four files for this assignment: "fileFreqAnalysis.py" for problem 1, "subEval.py" for problem 2, your "a5.pdf" or "a5.txt" for problem 3, and a README file (also in either PDF or plain text). All encipherment and decipherment in this assignment uses the simple substitution cipher.

Problem 1

Using code from the textbook as a starting point, create a Python module named "fileFreqAnalysis.py", containing functions named "freqDict" and "freqDecrypt". "freqDict" and "freqDecrypt" will be called as follows: **freqDict(f1,frequency), freqDecrypt(f1,f2,frequency)**

$f1$ will be a string that is the name of a .txt file, where the contents are a ciphertext enciphered with a simple substitution cipher. The cipher characters will be uppercase letters, as normal. $f2$ will be a string that is the name of a .txt file that should be produced by the function "freqDecrypt". The text file created should be in the same directory as your programs. $frequency$ will be a string of letters in decreasing frequency order. You can assume that every letter that occurs in $f1$ will be in frequency.

The "freqDict" function should perform frequency analysis on the entire text, using the frequency statistics from the textbook, and return a dictionary, whose **keys are the cipher characters**, with the **value for each key being the plaintext character** it is assigned to using frequency analysis. For example, the value of the most frequent cipher character should be 'E', and the value of the least frequent cipher character should be 'Z' if the parameter $frequency$ is ETAOIN.

The "freqDecrypt" function should perform frequency analysis on $f1$, the ciphertext file, **decipher it** using the resulting mapping (i.e. mapping all instances of the most frequent cipher character to 'E' if $frequency$ is ETAOIN, and so on; you should call your "freqDict" function to get the mapping), and **print the resulting decipherment to the file** $f2$, the specified output file-name.

Problem 2

There are two primary ways of evaluating a solution to a simple substitution cipher: *key accuracy* and *decipherment accuracy.* Key accuracy is the proportion of cipher character types in the alphabet which are mapped to their correct plaintext characters. Decipherment accuracy is the proportion of cipher character tokens in the ciphertext which are mapped to their correct plaintext characters.

Create a Python file named "subEval.py" which has a function named "evalFile"called as follows: **evalFile(f1,f2)**. Both parameters are strings that are the names of files. $f1$ is an imperfect plaintext that was deciphered by some method (i.e. frequency analysis) and $f2$ is the original plaintext. Your code should compare the two files, and **return a two-element tuple (keyAccuracy, deciphermentAccuracy)**, which compares the key and decipherment accuracy of $f1$. Both elements to be returned should be **floats**.

Your program should only count alphabetical characters in its decipherment evaluation and it should be case-insensitive.

Example: If $f1$ contains the plaintext "this is an example", $f2$ might contain "tsih ih an ezample" – the text in $f1$ was enciphered, and $f2$ contains an imperfect attempt to decipher it. This decipherment has a key accuracy of 8/11, since there are 11 character types, and three of them, 'h', 's', and 'x', were deciphered incorrectly; it also has a decipherment accuracy of 11/15, since it is 15 characters long, but only 11 character tokens in the decipherment are correct. Thus, the function should return the list [0.7272727272727273, 0.7333333333333333].

Problem 3

Short answer: You now have at least two different ways of hacking the simple substitution cipher: frequency analysis (deciphering the n^{th} most frequent cipher symbol into the n^{th} most frequent plaintext symbol), and the fully-automated dictionary-based method presented in the textbook, in simpleSubHacker.py.

Included with this assignment is a zip file with five plaintexts each with a copy that has been enciphered using the substitution cipher and a random key Some of these texts may not be in English).

Using your code from problem 1, decipher each of these texts using frequency analysis, and report the key accuracy and decipherment accuracy that frequency analysis obtains for each of the five texts. Repeat this procedure for all five ciphertexts using the solver from the textbook ("simpleSubHacker.py") and a substitution solver of your choice that you have found on the internet. **You must provide the URL for this solver** so that your results can be replicated and verified. You should present your results in a table like the one below:

Ciphertext	Frequency Analysis		Textbook Solver		Found Method	
	Key Acc.	Deciph. Acc.	Key Acc.	Deciph. Acc.	Key Acc.	Deciph. Acc.
alpha						
beta						
gamma						
delta						
epsilon						

1. Does the choice of key used to encrypt the text files affect the results of your evaluation? Explain.
2. Based on the data you have recorded for the provided texts, what conclusions can you make about the method of decipherment used by your program?
3. Some of the texts were not in English, how could you attempt to hack these encrypted messages?
4. Can you think of any changes that could you make to your program to improve its performance?

Include your accuracy measurements, your answers to the above questions, and the URL for the solver that you found on the internet in your "a5.pdf" or "a5.txt" file.