CMPUT 299, Winter 2018, Assignment 10

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 two files for this assignment: "decipher.py" and a README file (either PDF or plain text). **If your code requires any external modules or other files to run, include them in your submission as well. Your submission should be self-contained.**

Throughout the course, you have seen ciphers which apply substitutions to each letter in the plaintext individually, such as the simple substitution cipher. More recently, you have seen the Playfair cipher, which performs substitution operations on pairs of letters, or bigrams, for encryption and decryption.

The goal of this assignment is to use frequency analysis over bigram pairs to decrypt enciphered words encrypted with a bigram-substitution cipher, such as the Playfair cipher. The principle is the same as it was for the simple substitution cipher, only modified to handle bigrams rather than unigrams: given that the cipher algorithm always enciphers each bigram the same way, the most frequent bigrams of a large ciphertext should correspond to the most frequent bigrams in the English language.

You can begin by finding the most frequent bigrams of the encoded text. To do so, your program should count the number of non-overlapping bigrams in the ciphered text. For example, if the ciphered text is KILXBYOHWGPE, non-overlapping bigrams corresponding to this text are: KI, LX, BY, OH, WG, and PE. Then your program should find the most frequent bigrams in the English language. To do so, we have provided a large corpus that you should use. For example, if a word in the corpus is DOWN, then the bigrams corresponding to this text are: DO, OW, and WN. Note that this is different than the non-overlapping bigrams from the ciphered text. Make sure to count only the bigrams that are composed of letters. After that, your program should map the N most frequent bigrams in the corpus to the N most frequent bigrams in the ciphered text, where N is an optional command line parameter, with a default value of 3. The N most frequent bigrams from the corpus would replace the N most frequent bigrams from the ciphertext that are found in the ciphertext word. By this replacement, some letters of the ciphertext word will be revealed. That is, the ciphertext word would be partially decrypted. In order to completely decrypt the ciphertext word, your program should search for the probable words that fit the partially decrypted word in the provided dictionary.txt file.

In order to search for the words in the dictionary that match the pattern, you must use regular expressions. You should construct a regular expression by replacing each unknown letter in the ciphertext word with a dot, which stands for any arbitrary letter. For example if the partially deciphered word is KXtherinPZ, in which the capital letters are the ones that have not yet been revealed, the regular expression corresponding to this word is ..therin.. and if we look it up in the dictionary or dictionary, we would have these candidate words: ['gathering', 'withering'].

Because bigram substitution ciphers such as the Playfair cipher always results in a ciphertext that has an even number of symbols, we might sometimes also need to look for words that are shorter by one letter than the ciphertext word. You can figure out when to look for words that are shorter than the ciphertext word by one letter by understanding the mechanics of Playfair cipher algorithm (and you may assume that all bigram substitution ciphers have the same mechanics for ensuring their ciphertexts are of even length).

After finding the probable corpus words your program should output them on a single line (using a single space as a separator) in the order of their frequency in the provided corpus file. If words have the same frequency, output them in alphabetic order. There are words in the dictionary file which do

not occur in the corpus file (so their frequency in the corpus file is zero). These words should still be included in your output.

(Note: the enciphered text does not need to be deciphered: it is only used to collect ciphered bigram counts. Only the words provided as standard input need to be deciphered.)

In summary, your task is as follows:

1. Collect the count of all bigrams from the corpus file. You must use wells.txt as the corpus file; it is available on eClass.
2. Collect the count of non-overlapping bigrams from the ciphertext file. You must specify the ciphertext file as a command line option; the ciphertext file, playfair.txt, is available on eClass.
3. Map the N most common corpus bigrams to the N most common ciphertext bigrams. You must use the N value that is specified as a command line option; otherwise the value of N should default to 3.
4. You should read enciphered words from standard input. You may assume that these words consist of capital letters. A list of sample inputs for enciphered words have been given below. The input may consist of several lines.
5. Construct a regular expression by substituting all ciphertext bigrams in the ciphertext word with the corresponding corpus bigrams (if they are among the N most frequent).
6. Search for the partially-deciphered ciphertext word in the list of English words using the regular expressions. You must use the dictionary.txt file which is available on eClass. (Note: the word may have any number of characters, but the encoding will always have an even number of characters).
7. After finding the probable plaintext words your program should output them on a single line (using a single space as a separator)in the order of their frequency in the provided corpus file. If words have the same frequency, output them in the alphabetic order.

Some Important Details:

- To create the regular expression for finding the ciphertext word, replace only the ciphertext bigrams that are on the list of the N most frequent bigrams.
- Make sure to remove non-alphabetic characters from the corpus file before collecting bigram frequencies; otherwise your bigram counts may be incorrect.
- When counting the bigrams, bigrams with any white space as prefix/suffix are to be ignored.

An example run:

```
\$ python3 decipher.py playfair.txt
RNCZNC
health
RNBDKXIQ
heinlein heinous heinrich
BDRNZMDQ
inherent inheres inherit inherits
NCXZ
the that they this them then than thin thud thus thai thaw thea thor thug
```