CSE517 Assignment: A2_1.3

A well-known problem with substitution ciphers is that, in a long ciphertext, the frequency of encrypt(x) will be close to the frequency of x in plaintext. In natural languages, there is a lot of variance in different symbols' frequencies. (For words, "Zipf's Law" states that the probability of the rth most frequent word in a text corpus will have relative frequency proportional to 1 r .) Your task is to exploit this problem to decrypt the ciphertext we provide to you. The original plaintext is in all-upper-cased English. You may assume the space symbol, numerical digits, and any other non-alphabetic symbols are fixed. This implies that N = 26. We suggest that you automate the calculation and visualization of the symbol frequencies. Some manual search may be required since your ciphertext (and any plaintext you use to estimate English relative frequencies) will be finite, leading to variance in your estimates. You should submit your decrypted text.

In [38]:
```
import random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from math import sqrt
```

In [11]:
```
e=open(r'/Users/nehakardam/Documents/UWclasses /CSE NLP/A2/A2.encrypted
```

In [12]:
```
def convert_line_to_dict(input_line, char_dict):
    for char in input_line:
        if char in char_dict:
            char_dict[char] = char_dict.get(char) + 1
        else:
            if (ord(char) >= 65) and (ord(char) <= 90):
                char_dict[char] = 1

def convert_text_to_dict(file, char_dict):
    for line in file:
        convert_line_to_dict(line, char_dict)
```

In [13]:
```
encrypted_char_dict = {}
convert_text_to_dict(e, encrypted_char_dict)
encrypted_char_list = sorted(encrypted_char_dict.items(), key=lambda x:
encrypted_char_list
```

Out[13]:
```
[('O', 495),
 ('H', 357),
 ('Z', 313),
 ('L', 288),
 ('B', 283),
 ('F', 277),
 ('T', 266),
 ('X', 209),
 ('D', 182),
 ('V', 170),
 ('C', 159),
 ('Y', 126),
 ('I', 111),
 ('R', 97),
 ('N', 88),
 ('P', 84),
 ('U', 79),
 ('E', 76),
 ('G', 76),
 ('S', 73),
 ('K', 46),
 ('W', 42),
 ('M', 9),
 ('Q', 6),
 ('A', 5),
 ('J', 4)]
```

Train Corpus: https://www.gutenberg.org/files/65728/65728-0.txt (https://www.gutenberg.org/files/65728/65728-0.txt)

In [14]:
```
d=open(r'/Users/nehakardam/Documents/UWclasses /CSE NLP/A2/A2_train_cor
```

In [15]:
```
plain_char_dict = {}
convert_text_to_dict(d, plain_char_dict)
plain_char_list = sorted(plain_char_dict.items(), key=lambda x: x[1], r
plain_char_list
```

Out[15]:
```
[('E', 93834),
 ('T', 75541),
 ('O', 66465),
 ('I', 56202),
 ('R', 56152),
 ('N', 52641),
 ('A', 52318),
 ('S', 42572),
 ('H', 33268),
 ('D', 31694),
 ('L', 27999),
 ('U', 25138),
 ('C', 23638),
 ('G', 19864),
 ('M', 18199),
 ('P', 18062),
 ('Y', 17649),
 ('F', 16902),
 ('W', 14910),
 ('B', 12856),
 ('K', 7796),
 ('V', 5374),
 ('J', 3106),
 ('Q', 1586),
 ('X', 1242),
 ('Z', 62)]
```

In [16]:
```
converter_dict = {}
for i in range(26):
    converter_dict[encrypted_char_list[i][0]] =  plain_char_list[i][0]
# converter_dict['I'] = 'S'
converter_dict['F'] = 'S'
converter_dict['E'] = 'B'
converter_dict['N'] = 'U'
converter_dict['C'] = 'U'
converter_dict['B'] = 'A'
converter_dict['U'] = 'G'
converter_dict['T'] = 'N'
converter_dict['Y'] = 'M'
converter_dict['S'] = 'P'
converter_dict['I'] = 'D'
converter_dict['P'] = 'F'
converter_dict['K'] = 'V'
converter_dict['V'] = 'H'
converter_dict['X'] = 'R'
converter_dict['G'] = 'Y'
converter_dict['R'] = 'C'
converter_dict['D'] = 'L'
converter_dict['Q'] = 'Z'
converter_dict['W'] = 'K'
converter_dict['J'] = 'Q'
converter_dict['N'] = 'W'
# converter_dict['V'] = 'H'
converter_dict
```

Out[16]:
```
{'O': 'E',
 'H': 'T',
 'Z': 'O',
 'L': 'I',
 'B': 'A',
 'F': 'S',
 'T': 'N',
 'X': 'R',
 'D': 'L',
 'V': 'H',
 'C': 'U',
 'Y': 'M',
 'I': 'D',
 'R': 'C',
 'N': 'W',
 'P': 'F',
 'U': 'G',
 'E': 'B',
 'G': 'Y',
 'S': 'P',
 'K': 'V',
 'W': 'K',
 'M': 'J',
 'Q': 'Z',
 'A': 'X',
 'J': 'Q'}
```

In [17]:
```
e=open(r'/Users/nehakardam/Documents/UWclasses /CSE NLP/A2/A2.encrypted
o=open(r'/Users/nehakardam/Documents/UWclasses /CSE NLP/A2/A2.decrypted

def decrypt_line(input_line):
    output_line = ""
    for char in input_line:
        if (ord(char) >= 65) and (ord(char) <= 90):
            output_line = output_line + converter_dict[char]
        else:
            output_line = output_line + char
    return output_line

def decrypt_text(file):
    for line in file:
        nline = decrypt_line(line)
        o.write(nline)
    o.close()

decrypt_text(e)
```

In [21]:
```
total_count = 0
encrypted_char_list_with_freq = []
for element in encrypted_char_list:
    total_count = total_count + element[1]
for element in encrypted_char_list:
    encrypted_char_list_with_freq.append((converter_dict[element[0]], e
encrypted_char_list_with_freq
```

Out[21]:
```
[('E', 0.1262433052792655),
 ('T', 0.09104820198928845),
 ('O', 0.07982657485335373),
 ('I', 0.07345065034429993),
 ('A', 0.07217546544248916),
 ('S', 0.07064524356031625),
 ('N', 0.06783983677633257),
 ('R', 0.05330272889568988),
 ('L', 0.04641673042591176),
 ('H', 0.04335628666156593),
 ('U', 0.04055087987758225),
 ('M', 0.032134659525631215),
 ('D', 0.028309104820198928),
 ('C', 0.024738587095128793),
 ('W', 0.022443254271869422),
 ('F', 0.02142310635042081),
 ('G', 0.020147921448610048),
 ('B', 0.01938281050752359),
 ('Y', 0.01938281050752359),
 ('P', 0.018617699566437135),
 ('V', 0.011731701096659015),
 ('K', 0.010711553175210406),
 ('J', 0.0022953328232593728),
 ('Z', 0.001530221882172915),
 ('X', 0.0012751849018107625),
 ('Q', 0.00102014792144861)]
```

In [34]:
```
encrypter_dict = {v: k for k, v in converter_dict.items()}
encrypter_dict
```

Out[34]:
```
{'E': 'O',
 'T': 'H',
 'O': 'Z',
 'I': 'L',
 'A': 'B',
 'S': 'F',
 'N': 'T',
 'R': 'X',
 'L': 'D',
 'H': 'V',
 'U': 'C',
 'M': 'Y',
 'D': 'I',
 'C': 'R',
 'W': 'N',
 'F': 'P',
 'G': 'U',
 'B': 'E',
 'Y': 'G',
 'P': 'S',
 'V': 'K',
 'K': 'W',
 'J': 'M',
 'Z': 'Q',
 'X': 'A',
 'Q': 'J'}
```

Part 1.5: Lets add more symbols '@', '#', '$', '%', '~', '^'

|E U T| = 26 + 6 (new ones) = 32

Most frequent symbols: Number of symbols for 'E' = 3 Number of symbols for 'T' = 2 Number of symbols for 'O' = 2 Number of symbols for 'I' = 2 Number of symbols for 'A' = 2

In [35]:
```
encrypter_dict['E'] = ('O', '@', '#')
encrypter_dict['T'] = ('H', '%')
encrypter_dict['O'] = ('Z', '~')
encrypter_dict['I'] = ('L', '^')
encrypter_dict['A'] = ('B', '$')
```

In [36]:
```
encrypter_dict
```

Out[36]:
```
{'E': ('O', '@', '#'),
 'T': ('H', '%'),
 'O': ('Z', '~'),
 'I': ('L', '^'),
 'A': ('B', '$'),
 'S': 'F',
 'N': 'T',
 'R': 'X',
 'L': 'D',
 'H': 'V',
 'U': 'C',
 'M': 'Y',
 'D': 'I',
 'C': 'R',
 'W': 'N',
 'F': 'P',
 'G': 'U',
 'B': 'E',
 'Y': 'G',
 'P': 'S',
 'V': 'K',
 'K': 'W',
 'J': 'M',
 'Z': 'Q',
 'X': 'A',
 'Q': 'J'}
```

In [45]:
```
e=open(r'/Users/nehakardam/Documents/UWclasses /CSE NLP/A2/A2.decrypted
o=open(r'/Users/nehakardam/Documents/UWclasses /CSE NLP/A2/A2.new_encry
def encrypt_line(input_line):
    output_line = ""
    for char in input_line:
        if (ord(char) >= 65) and (ord(char) <= 90):
            if type(encrypter_dict['E']) is tuple:
                output_line = output_line + random.choice(encrypter_dic
            else:
                output_line = output_line + encrypter_dict[char]
        else:
            output_line = output_line + char
    return output_line

def encrypt_text(file):
    for line in file:
        nline = encrypt_line(line)
        o.write(nline)
    o.close()

encrypt_text(e)
```